Guess the Elo, & More!

Applied Machine Learning for Chess Game & -Player Characterization

NFYK20002U Final Project

Aidan James Sudler (vhr382)
Daniel Emre Fogh (qpd797)
Elias Ben Slama (vfd881)
Villads Hoff Søegaard (tjq587)

Our dataset

Lichess's database, https://database.lichess.org/

- Approx. 10M-100M games per month since 2013
- Nearly 18 TB of data (we use subsets!)

One game:

Metadata

SAN string

```
[Event "Rated Bullet tournament https://lichess.org/tournament/yc1WW2Ox"]
[Site "https://lichess.org/PpwPOZMq"]
[Date "2017.04.01"]
[Round "-"]
[White "Abbot"]
[Black "Costello"]
[Result "0-1"]
[UTCDate "2017.04.01"]
[UTCTime "11:32:01"]
[WhiteElo "2100"]
[BlackElo "2000"]
[WhiteRatingDiff "-4"]
[BlackRatingDiff "+1"]
[WhiteTitle "FM"]
[ECO "B30"]
[Opening "Sicilian Defense: Old Sicilian"]
[TimeControl "300+0"]
[Termination "Time forfeit"]

1. e4 { [%eval 0.17] [%clk 0:00:30] } 1... c5 { [%eval 0.19] [%clk 0:00:30] }
2. Nf3 { [%eval 0.25] [%clk 0:00:29] } 2... Nc6 { [%eval 0.33] [%clk 0:00:30] }
3. Bc4 { [%eval -0.13] [%clk 0:00:28] } 3... e6 { [%eval -0.04] [%clk 0:00:30] }
4. c3 { [%eval -0.4] [%clk 0:00:27] } 4... b5? { [%eval 1.18] [%clk 0:00:30] }
5. Bb3?! { [%eval 0.21] [%clk 0:00:26] } 5... c4 { [%eval 0.32] [%clk 0:00:29] }
6. Bc2 { [%eval 0.2] [%clk 0:00:25] } 6... a5 { [%eval 0.6] [%clk 0:00:29] }
7. d4 { [%eval 0.29] [%clk 0:00:23] } 7... cxd3 { [%eval 0.6] [%clk 0:00:27] }
8. Qxd3 { [%eval 0.12] [%clk 0:00:22] } 8... Nf6 { [%eval 0.52] [%clk 0:00:26] }
9. e5 { [%eval 0.39] [%clk 0:00:21] } 9... Nd5 { [%eval 0.45] [%clk 0:00:25] }
10. Bg5?! { [%eval -0.44] [%clk 0:00:18] } 10... Qc7 { [%eval -0.12] [%clk 0:00:23] }
11. Nbd2?? { [%eval -3.15] [%clk 0:00:14] } 11... h6 { [%eval -2.99] [%clk 0:00:23] }
12. Bh4 { [%eval -3.0] [%clk 0:00:11] } 12... Ba6? { [%eval -0.12] [%clk 0:00:23] }
13. b3?? { [%eval -4.14] [%clk 0:00:02] } 13... Nf4? { [%eval -2.73] [%clk 0:00:21] } 0-1
```

Elo rating system

Measures a player's *relative* skill level, **updating up or down** based on game outcomes relative to the opponent's rating.

Depends **only** on:

- Game outcome (who won/lost?)
- Players' initial Elo

Elo range	Title / Category
2800+	*World champ.*
2500-2799	*Grandmaster*
2200-2499	*Master*
2000-2199	*Expert*
1600-1999	*Class A/B (Club / Tournament players)*
1200-1599	*Class C/D (Hobbyists)*
800-1199	*Novice*
< 800	*Beginner*

What problems can we define?

- **Guessing Elo – a *regression* problem**: Given Elo depends only on initial Elo and game outcome, can we train a model to recognize the players' Elo based on the game itself?

- **Playstyle *clustering***: Do players have a particular style, and can these styles be clustered? If so, is there correlation between a player's cluster and their Elo?

- **Blunder *classification***: Given the player's previous and current position on the board, can we predict if they will make a "blunder" within a 5-move window?

The setting of Chess can accommodate all three main ML problem types!

Regression with BDT – Guessing the Elo

Problem with naive approach

- Elo is the center of a gaussian distribution, hard to guess an Elo from one game

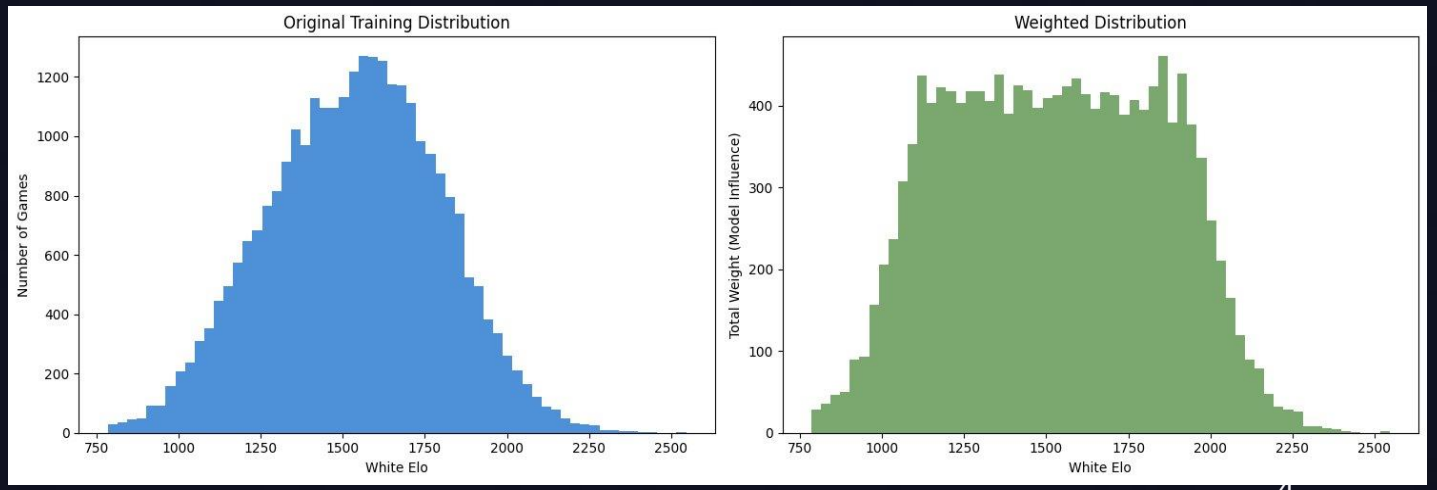
Data proces

- Feature extraction
- Lazy model guesses the average
- Weighting the distribution
- Learning from the outer points
- Optimizing for MAE
- Conflicting, how to balance?

SAN string

1. e4 e5
2. Nf3 Nc6
3. Bb5 a6

Numerical Summary Features (42)

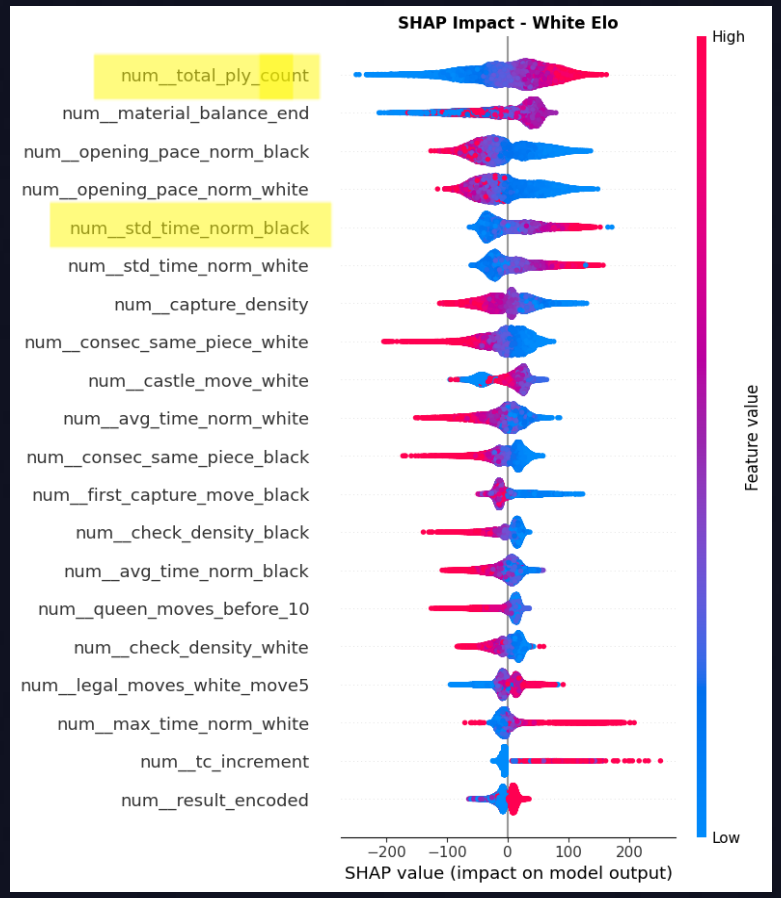


Regression with BDT

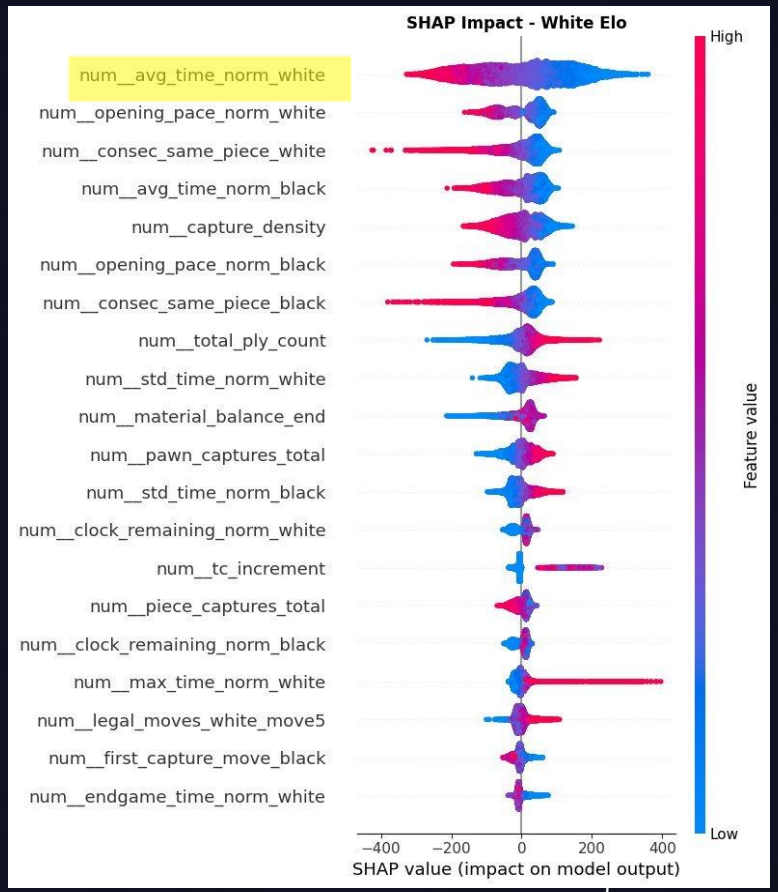
Data proces

- Shap – why do we sort for time control?
- Bullet (60 second) vs Rapid (10 min) games

10 min games



60s games



Regression with BDT – Guessing the Elo

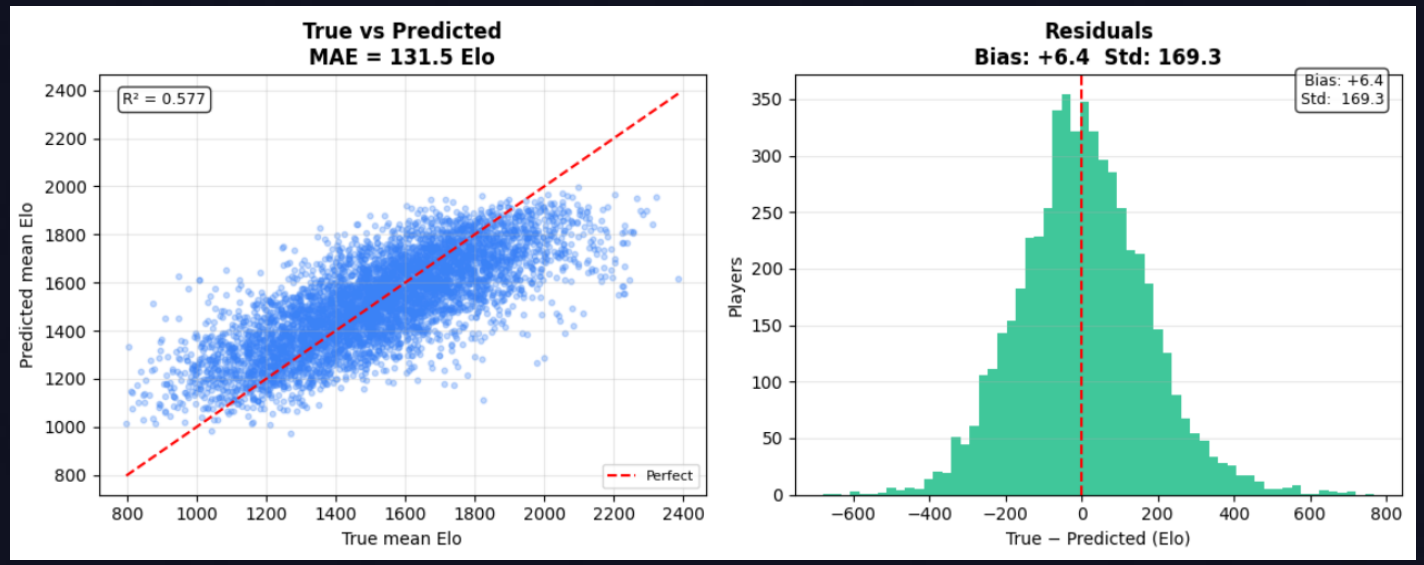
Solution

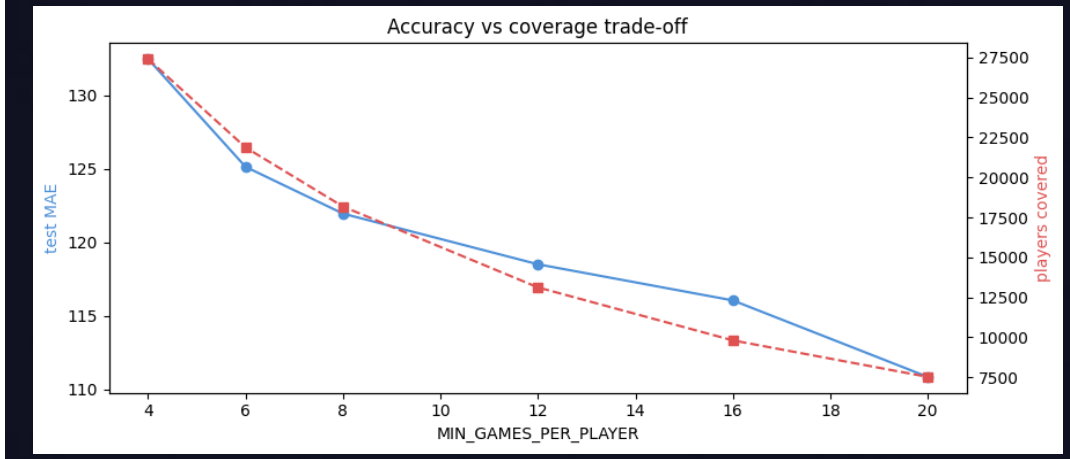
- Aggregate per player

Limits dataset...
2 million to 300k
to 30 k

	min_games	players	test_MAE
0	4	27408	132.5
1	6	21878	125.1
2	8	18181	121.9
3	12	13126	118.5
4	16	9812	116.0
5	20	7528	110.8

Model trained on average features per player

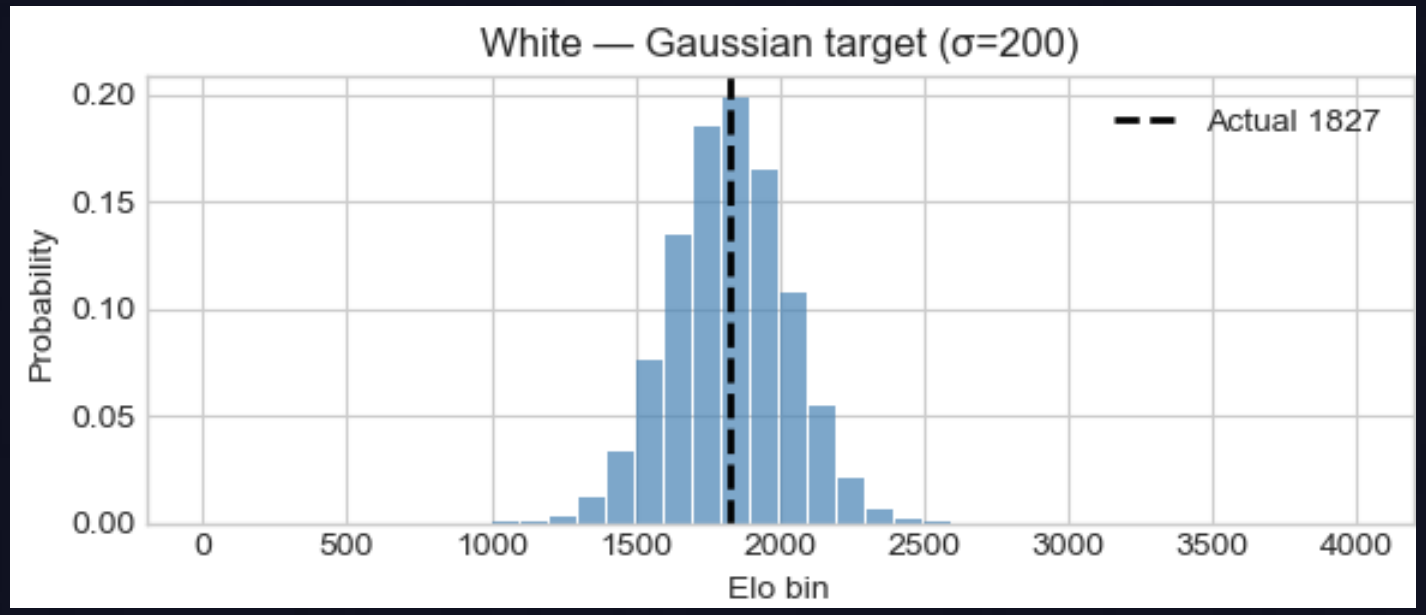




Predicting a Distribution, Not a Number

The problem with a single number: A player rated 1600 does not always play like a 1600

What we do instead: For every player we convert their Elo into a Gaussian bell curve centered on their true rating, spread across 40 rating bins → Loss Function!

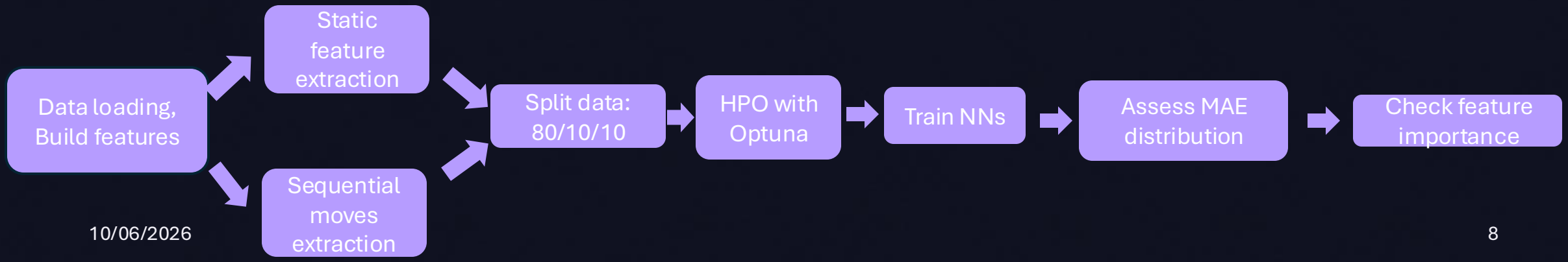


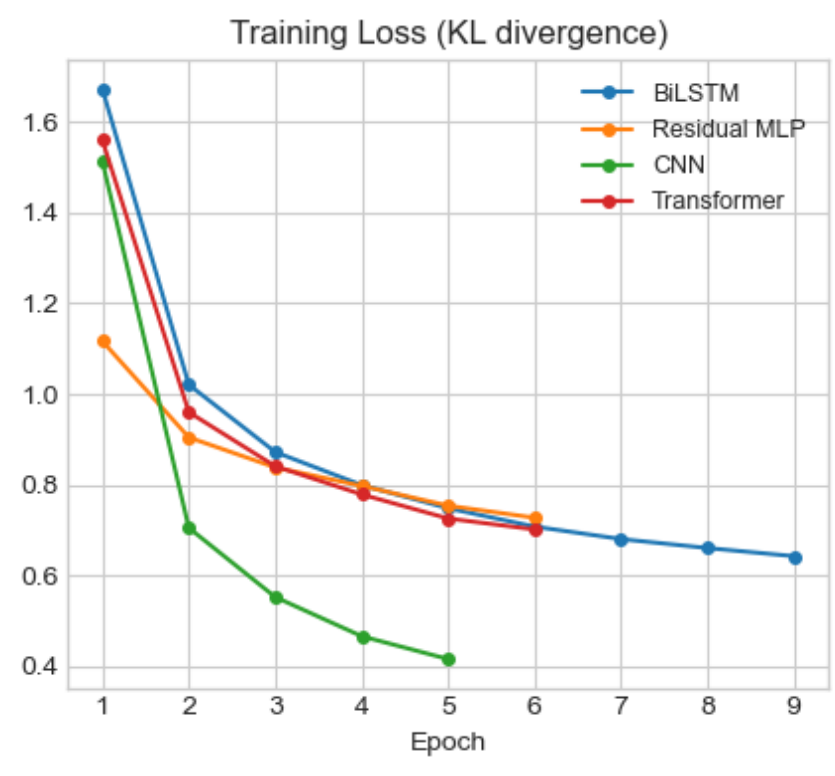
How we get back to a number: Once the model predicts a distribution, we recover a point estimate by taking the weighted average of all 40 bin centres

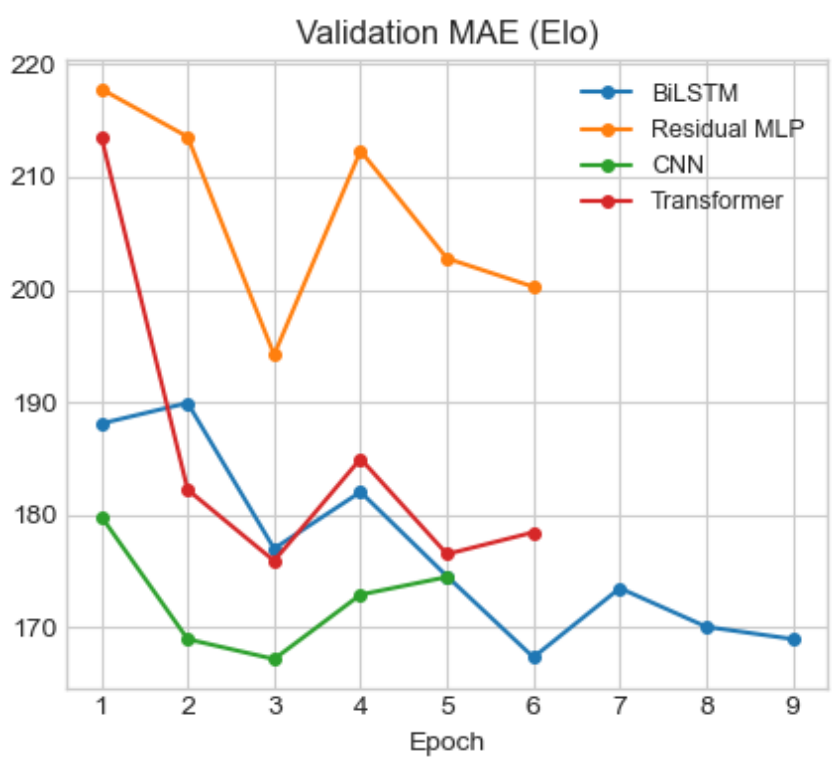
Neural Networks and predicting ELO

Model	Sequence Processing
BiLSTM	Two-layer bidirectional LSTM with soft attention
Residual MLP	No sequences — pure extracted features
1D CNN	Three parallel conv branches (kernels 3, 5, 7)
Transformer	Multi-head self-attention encoder

WORKFLOW







124.000 - 10 min games

CNN: MAE 168 - converged fastest (5 epochs)

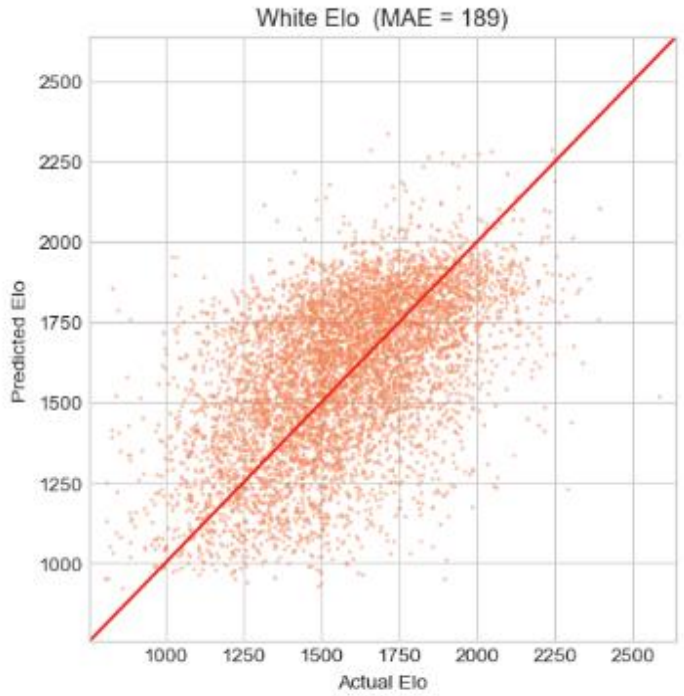
BiLSTM: MAE 165 - best but took 9 epochs

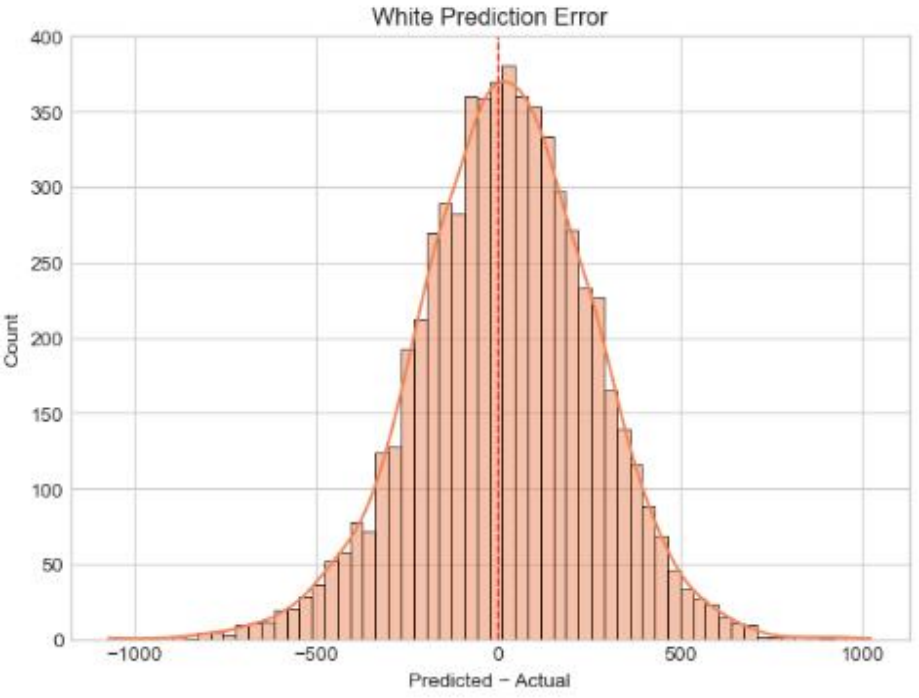
Transformer: MAE 173 - competitive but less stable val MAE

Residual MLP: MAE 189 - no sequence branch, solid baseline

Best model:

BiLSTM

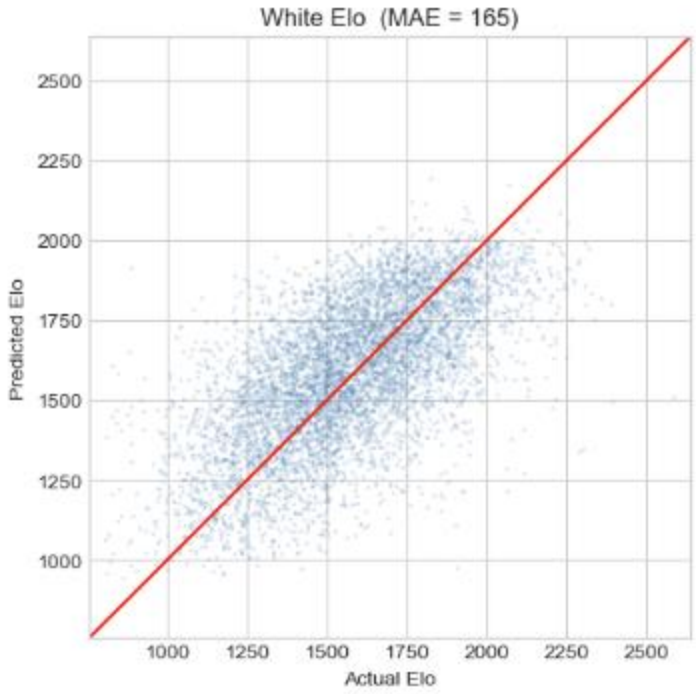


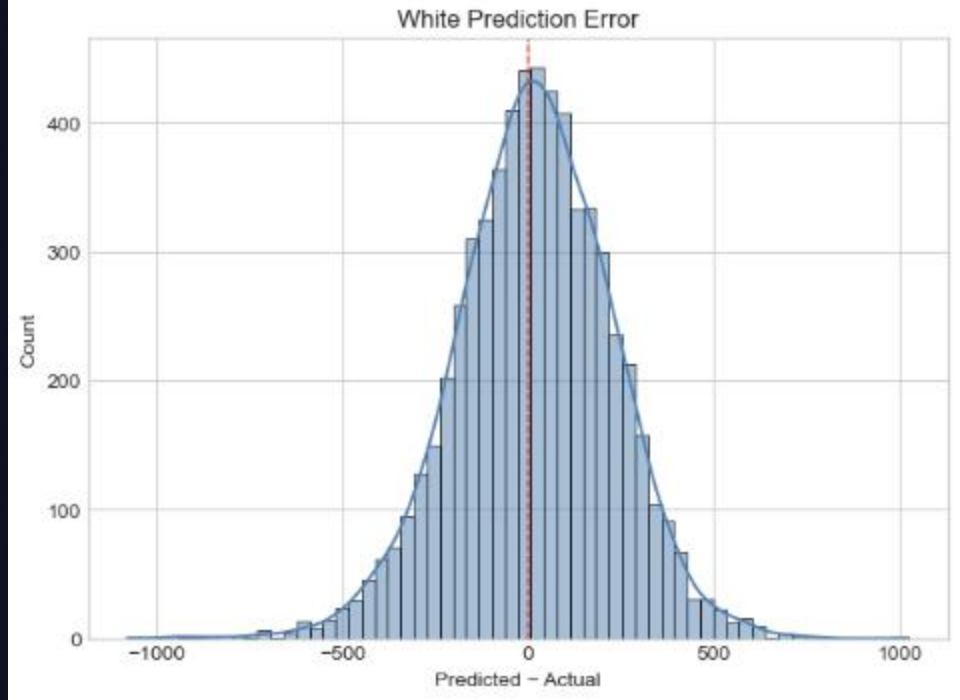


Prediction on test dataset

Worst vs. Best model

240 Std 210





The 3 N's: Did the Neural Network, Network well?

BiLSTM slightly better MAE than CNN
165 vs 168

Model	Total Time
BiLSTM	32m 7s
Residual MLP	3m 19s
CNN	4m 19s
Transformer	18m 40s

Challenges:

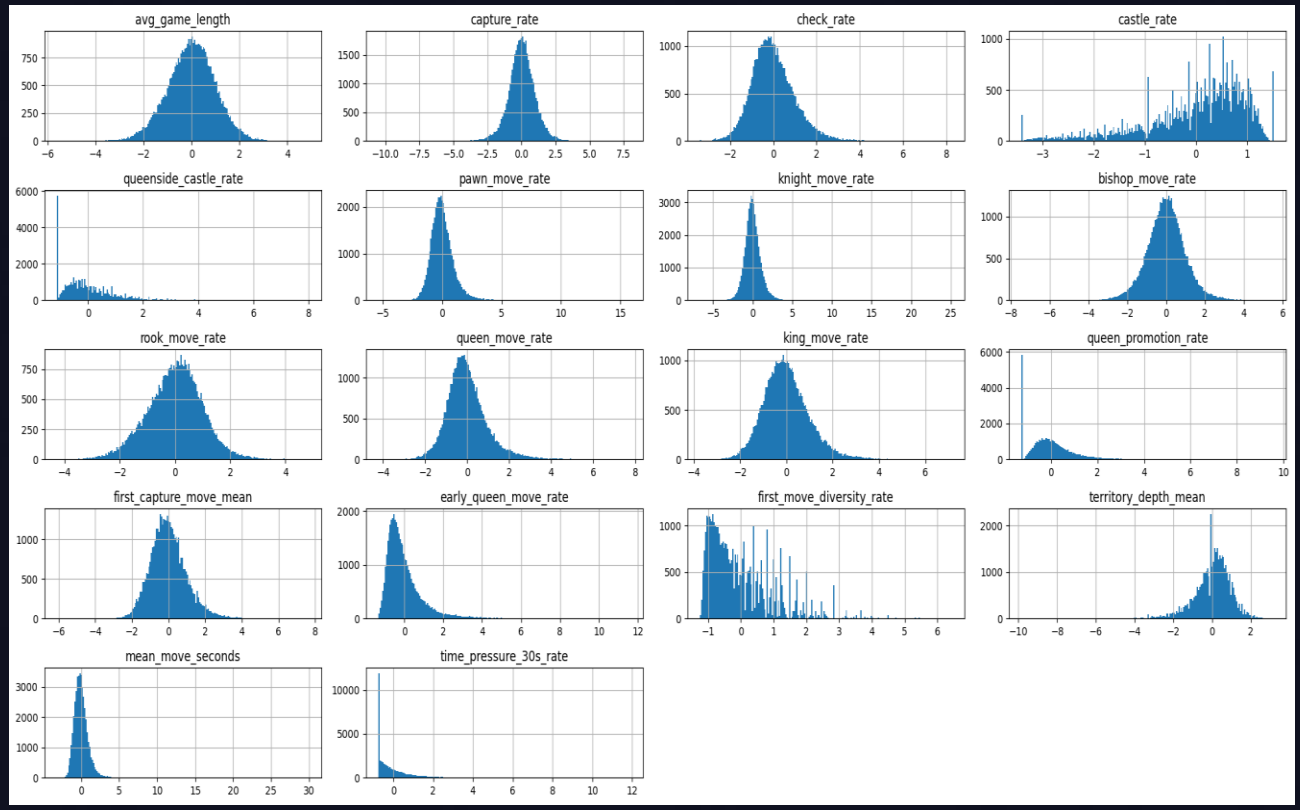
- HPO (Dropout)
- Data size vs. Going home
- Doing all time controls vs. Single

Playstyle clustering

- Data from single month (Sep. 2017) = 12.6M games.

- Parse game-data to player-data.
- Filter for:
  - **Time-control**: only *rapid* (10 min) games.
  - **Nr. games**: Only players w. > 10 games.

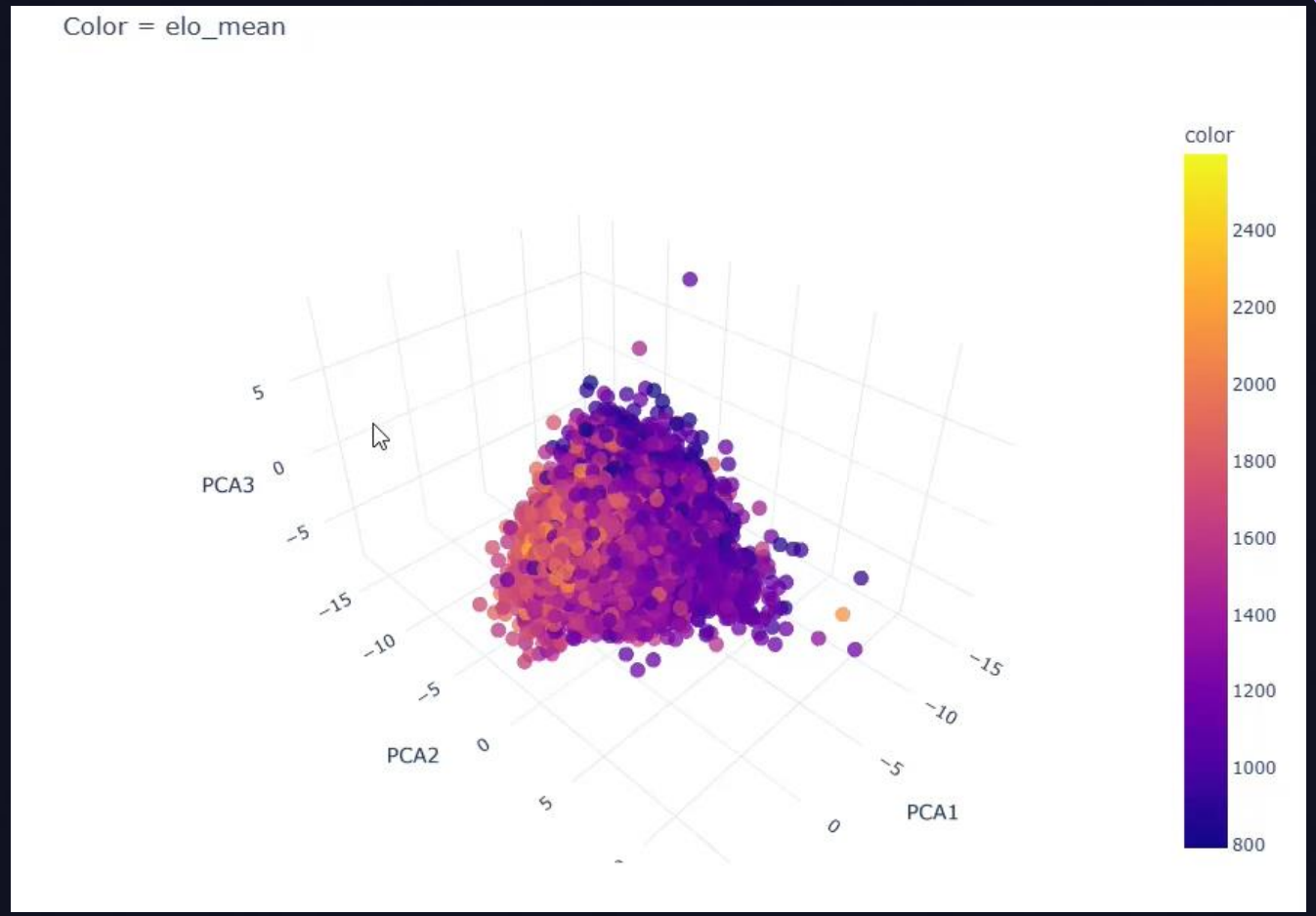
Leaves us with:
- 42K players (rows) described by
- 18 summary features (cols)

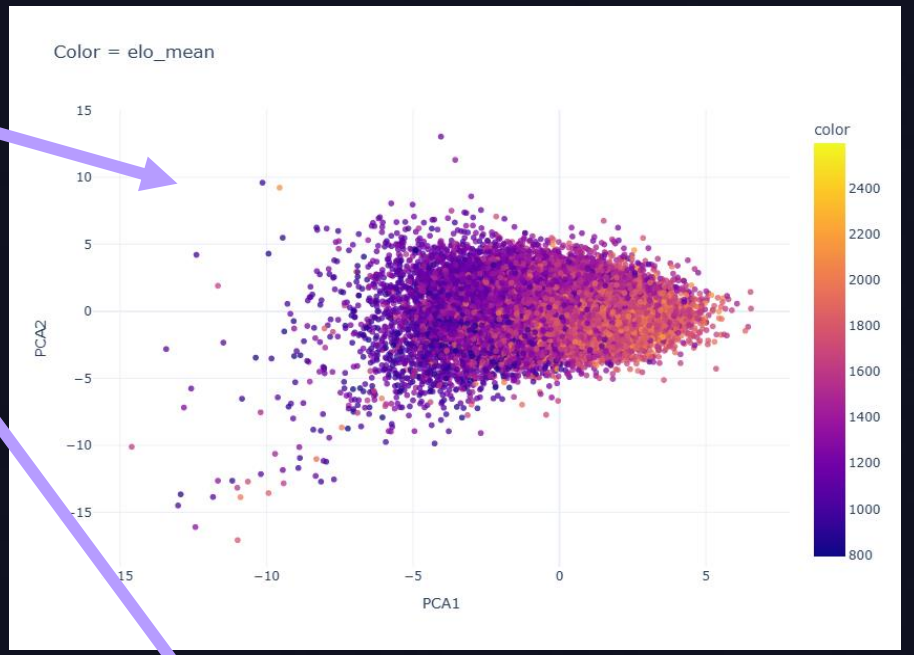


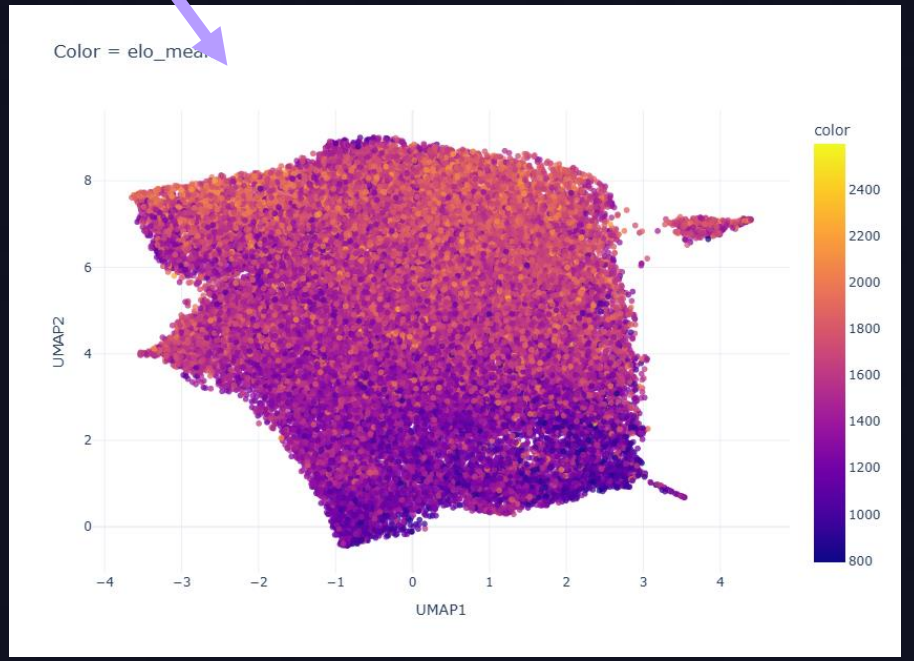
Plots of raw features

PCA

UMAP







KMeans *k* sweep

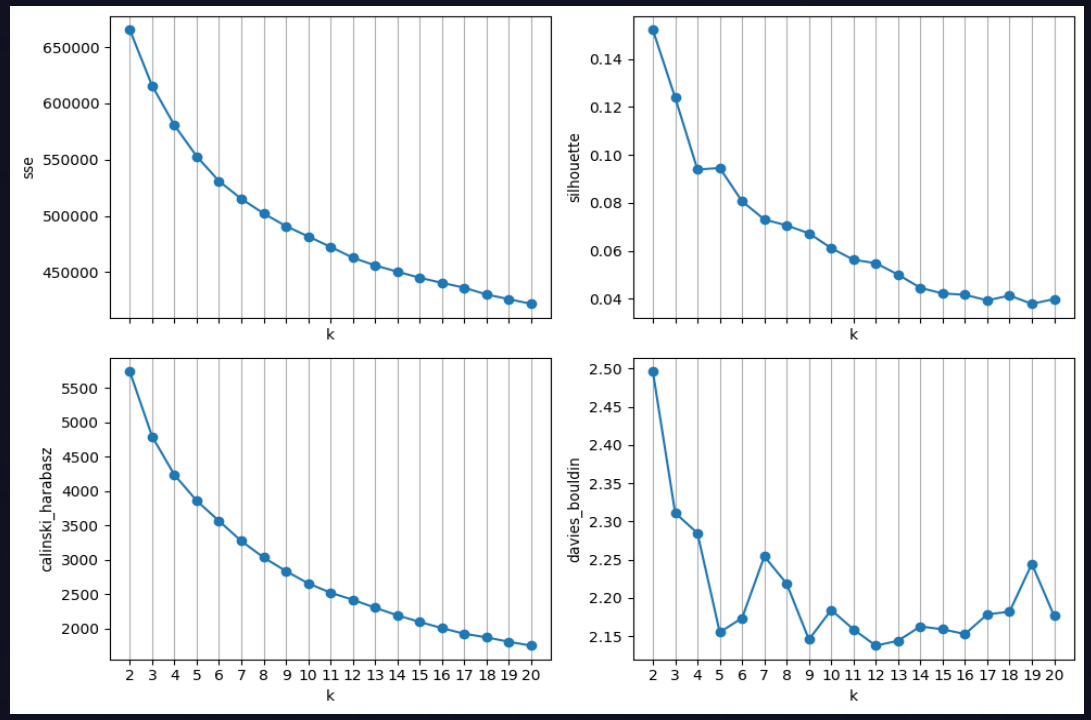
- No clear *k* vs. SSE elbow → reflects continuous data.
- Tried many HPs – no change.
- Overall inconclusive.

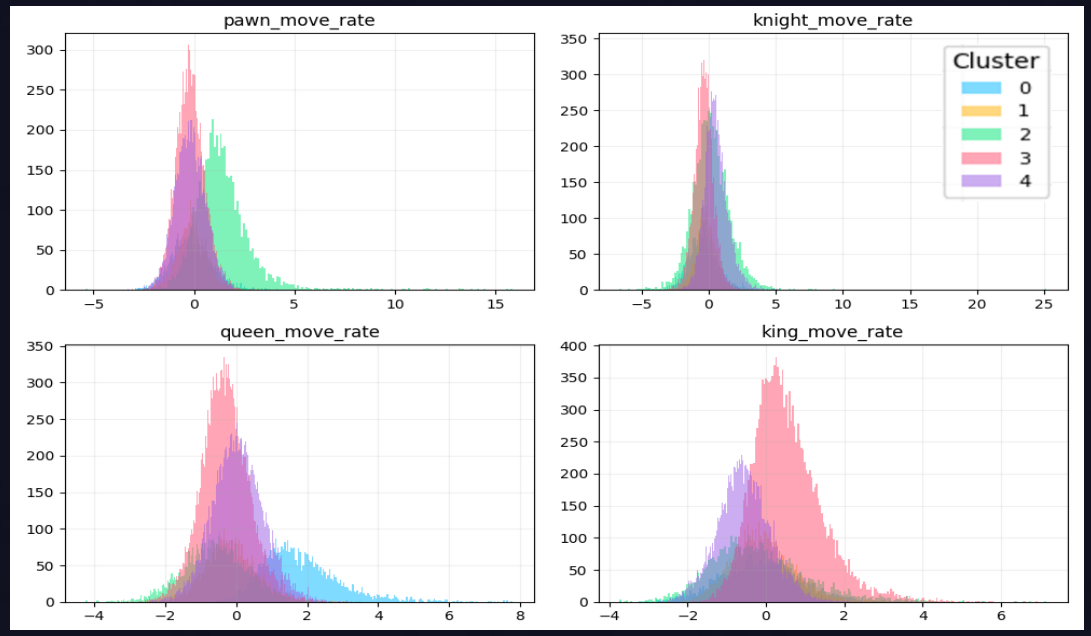
Had clusters been found, path forward would be:

- What feature creates biggest separation?
- Cluster-cluster separation (Cohen's *d*)
- Within-cluster similarity.
- Cluster-Elo correlation.

Conclusion: Data is *not* clustered, but shows structure - players lie on Elo gradient. Separations only exist in nonlinear embedding.

Makes intuitive sense?





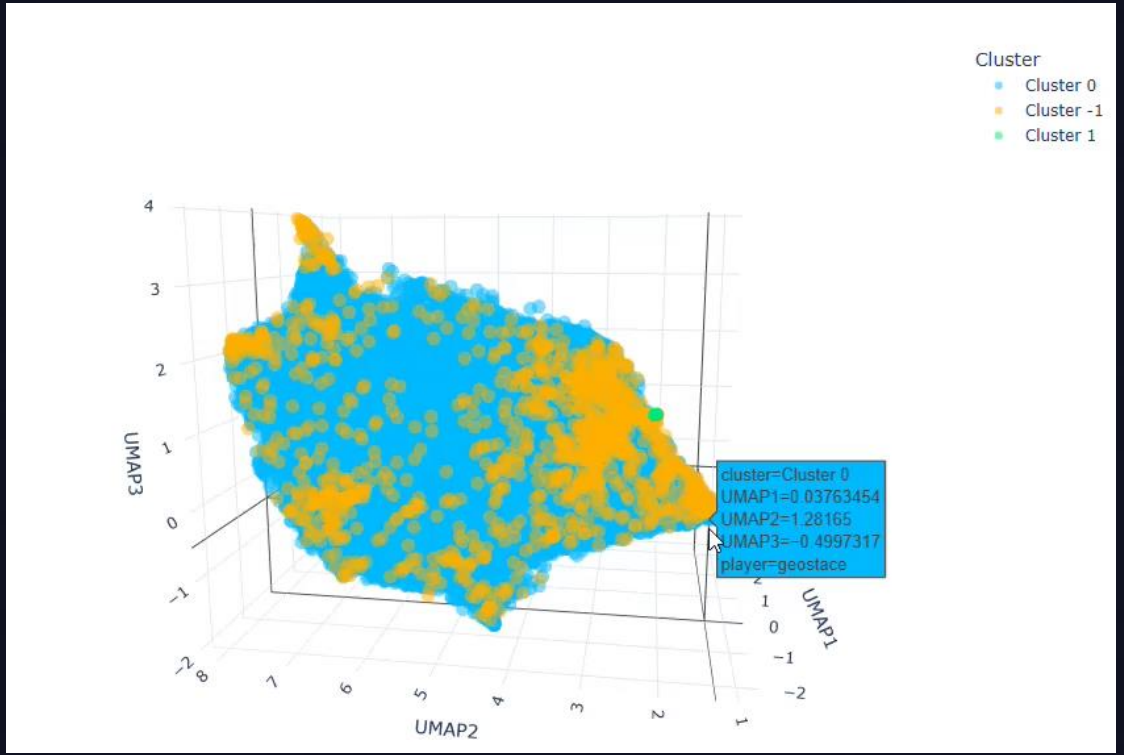
(Only showing 4 of 18)

Also tried:

- **AutoEncoder**: Bad. See Appendix.
- **DBSCAN**: Randomized HPO. Best model gives same results as Kmeans, although...

Identified 8 players with suspicious feature distributions...

So we checked their Lichess accounts...




```
{"eps":4.1792478216,
 "min_samples":10,
 "metric":"euclidean"}
```


Cluster	Counts
0	40638
1	8
-1 (Noise)	1357

StivYunior_Kesugihan

Kartaguss

akselsb

ArturIM

Makar_Batulin

912 Tournament Points | 0 Studies | 0 Forum Posts

The Lichess Terms of Service were violated on this account

Russia

Member since Jan 26, 2017

Active 8 years ago

Time spent playing: 15 days, 3 hours

Time featured on TV: 6 minutes

400 Elo increase in a month?

Not sayin' anything... but...

AmateurPL

473 Tournament Points | 1 Study | 0 Forum Posts

Watch | Challeng

1M 3M 6M YTD 1Y ALL

2500
2000
1000
500

2017 2019 2021 2023 2025

"Blunder detection" (classification problem)

A blunder is a move that severely weakens a player's position. Quantifiable: **≥ 2.00 pawn loss** from engine eval. **TASK:** Using BDTs, can we predict whether a player will blunder in a **5-move window** from their current position?

Exploratory dataset: **100,000 game** sample of Lichess games containing engine evaluation data from May 2017

Each turn as row -> **4,720,769 rows**

X: features related to a player's current position and move history (e.g. `total_material_current`, `clock_seconds_current`, `current_pawn_loss`, ...) (**37 or 50 total x 4,720,769 rows**)
y: `is_blunder` \in [0,1] (score)

<u>Key considerations</u>

1. Data balance:
   - ca. 10% of rows contain blunders,
   - ca. 32% of 5-move windows contain blunders.

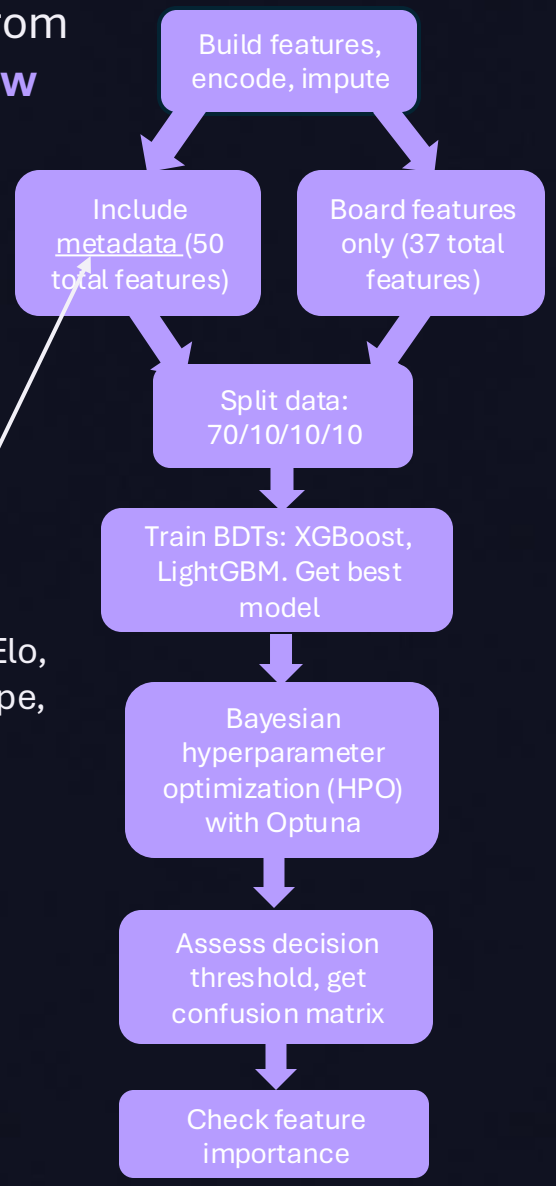
2. Encoding of non-numerical features:

<u>Key solutions</u>

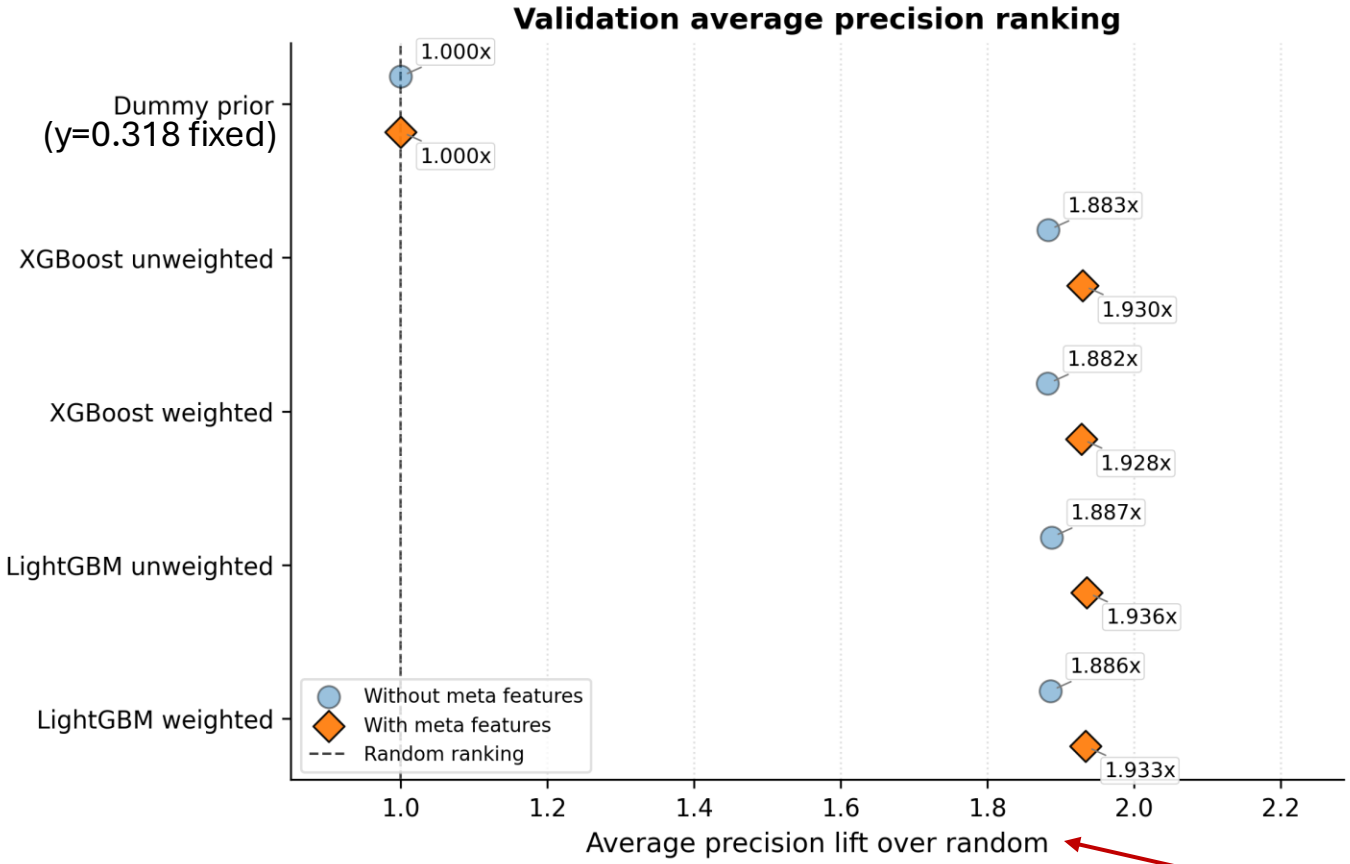
1. Don't use BCE as training target. Weigh inputs.

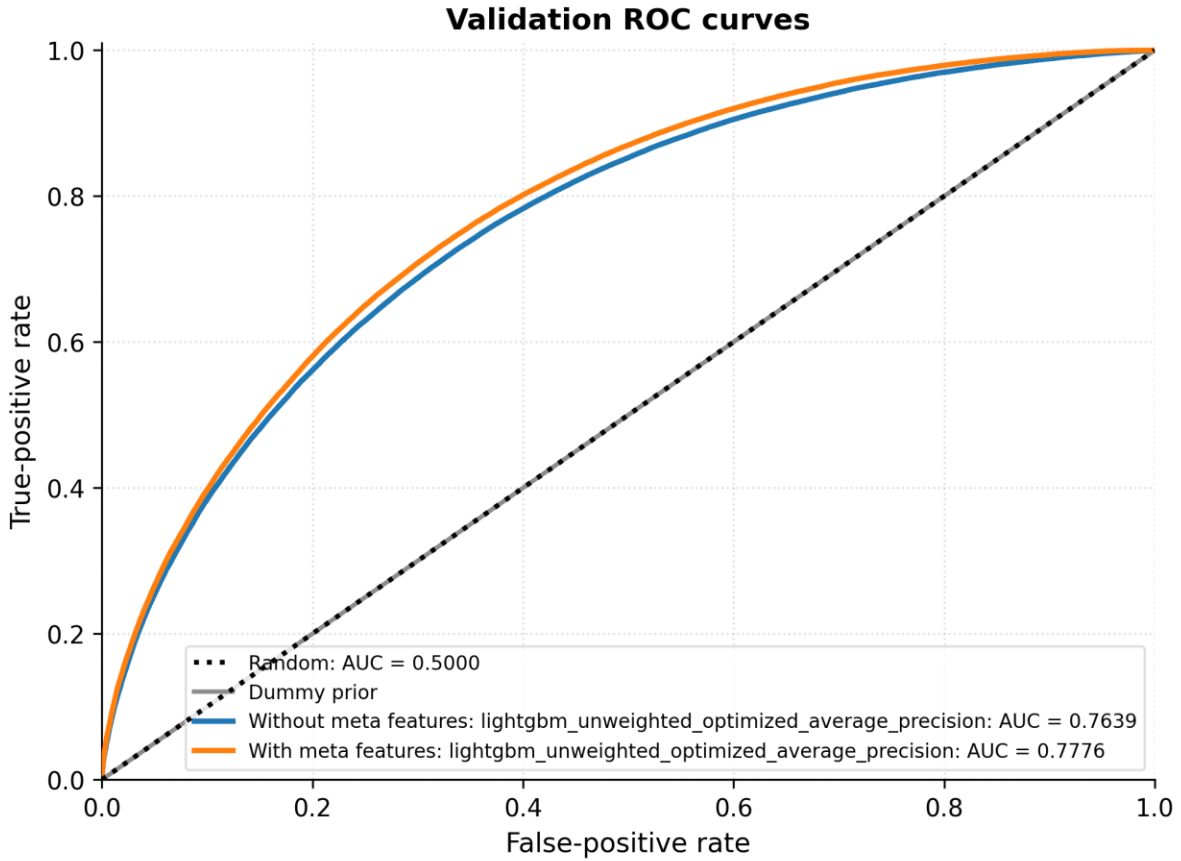
2. One-hot encoding.

<u>WORKFLOW</u>

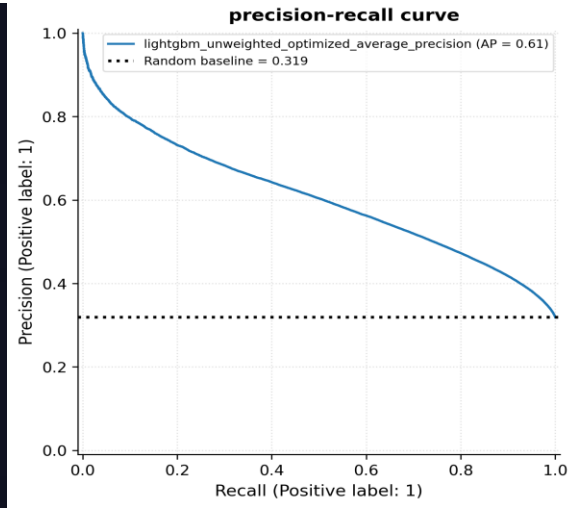


Validation comparison





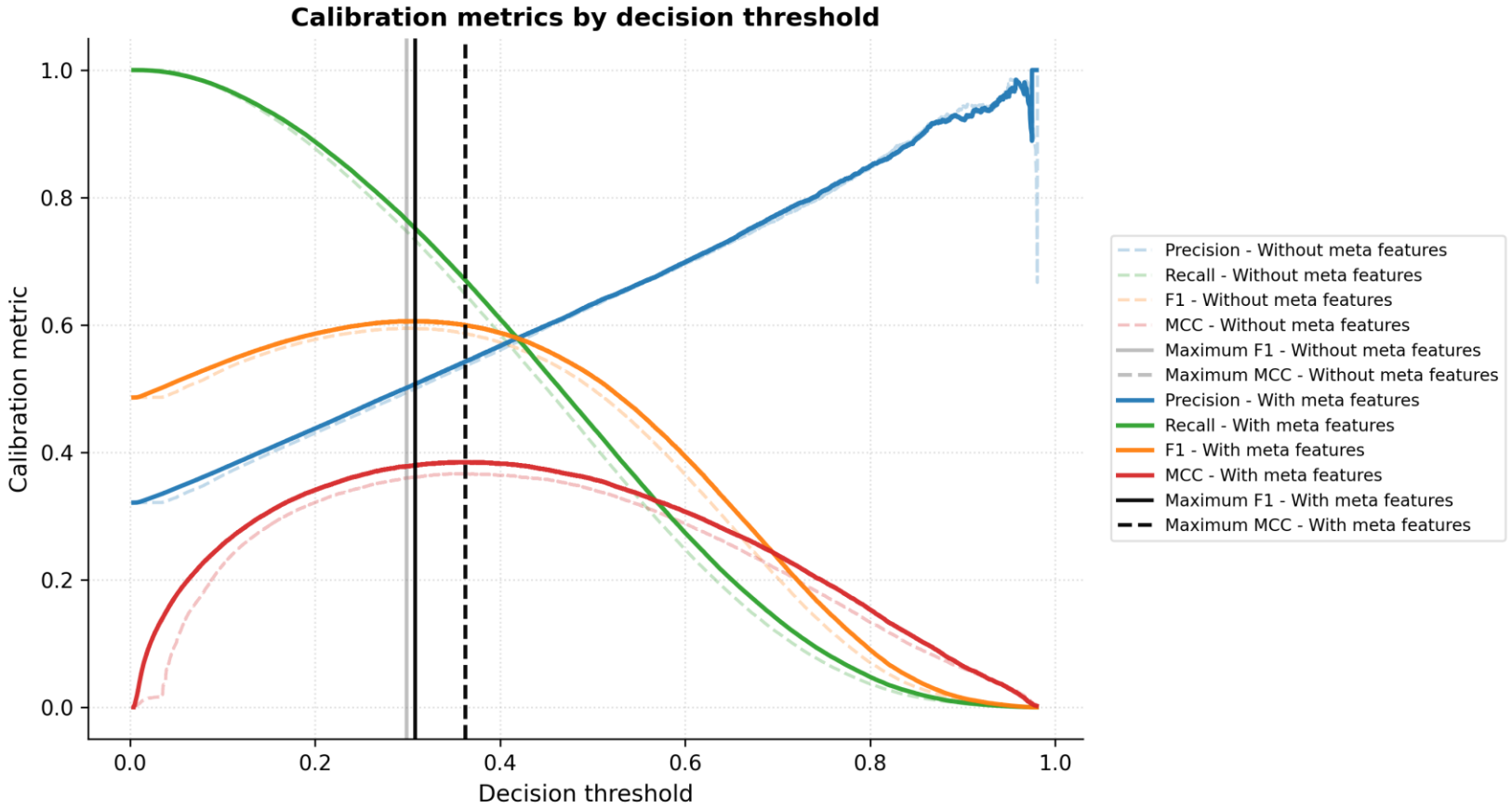
BDTs are trained to maximize **average precision (AP)**, which is the area under the precision-recall curve. Higher is better!

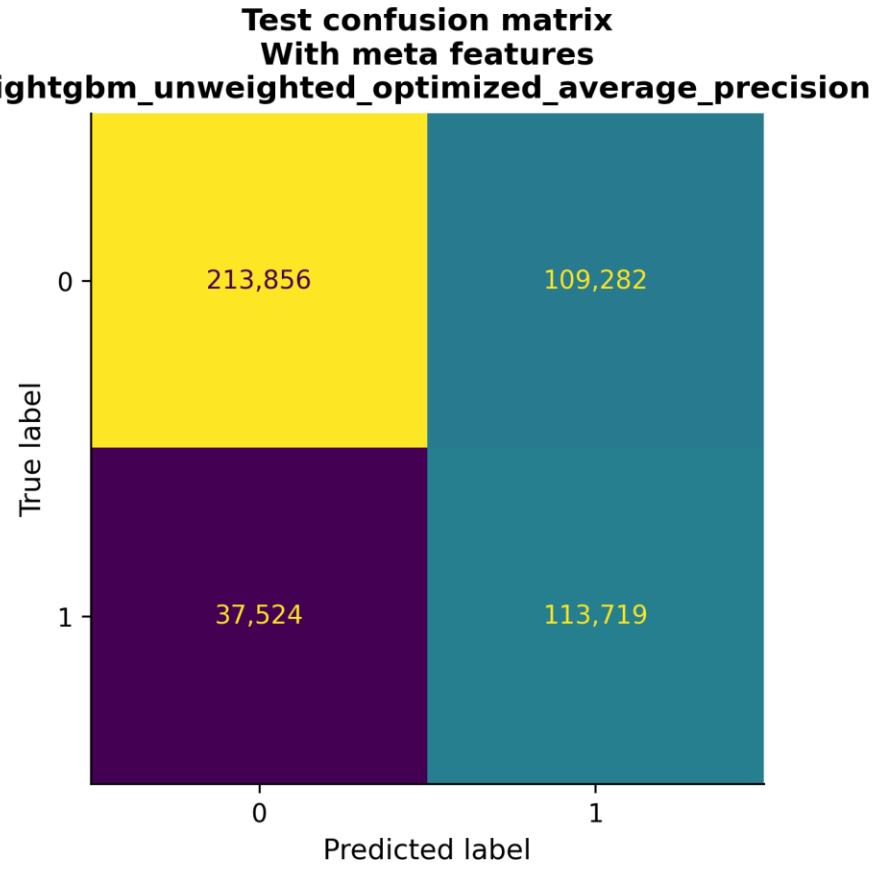


$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{AP lift} = \frac{\text{AP}_{\text{BDT}}}{\text{AP}_{\text{dummy}}}$$





$$F1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

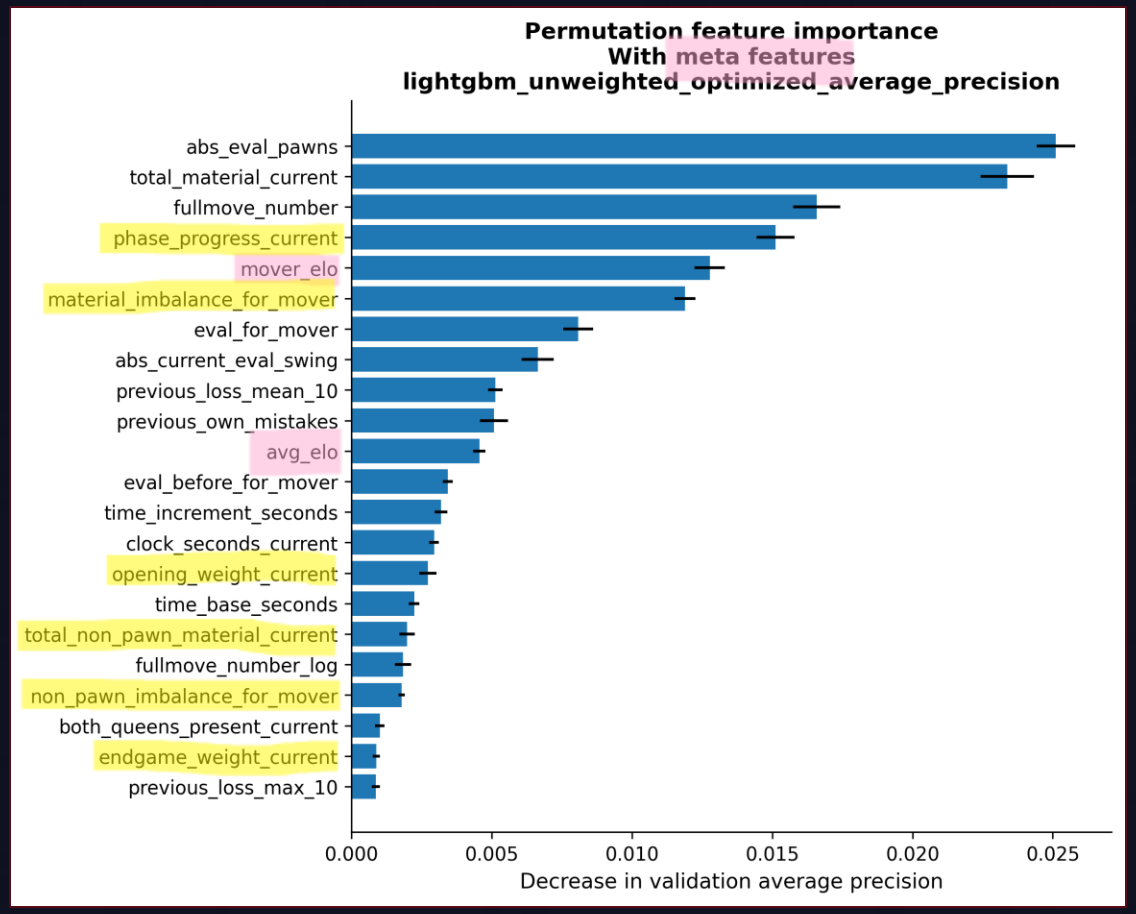
One of the commonly used measures of separation the MCC, which (in this case) is the Pearson ϱ, and related to the ChiSquare:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

For this confusion matrix, we used the BDT trained on regular + meta features and selected the decision threshold based on the best F1 score (threshold = 0.308, F1 = 0.606).

More false positives than false negatives because threshold is recall-heavy.

Permutation importance, takeaways



Having a player skill category gives more predictive power (Elo rating)

Only trained on a subset of games from May 2017, would have been nice to explore more data if time allowed it

BDT implementation only, no NN:(

Redundancy of (engineered) features?

"Blunder" definition is somewhat arbitrary here, how would results change for different blunder levels?

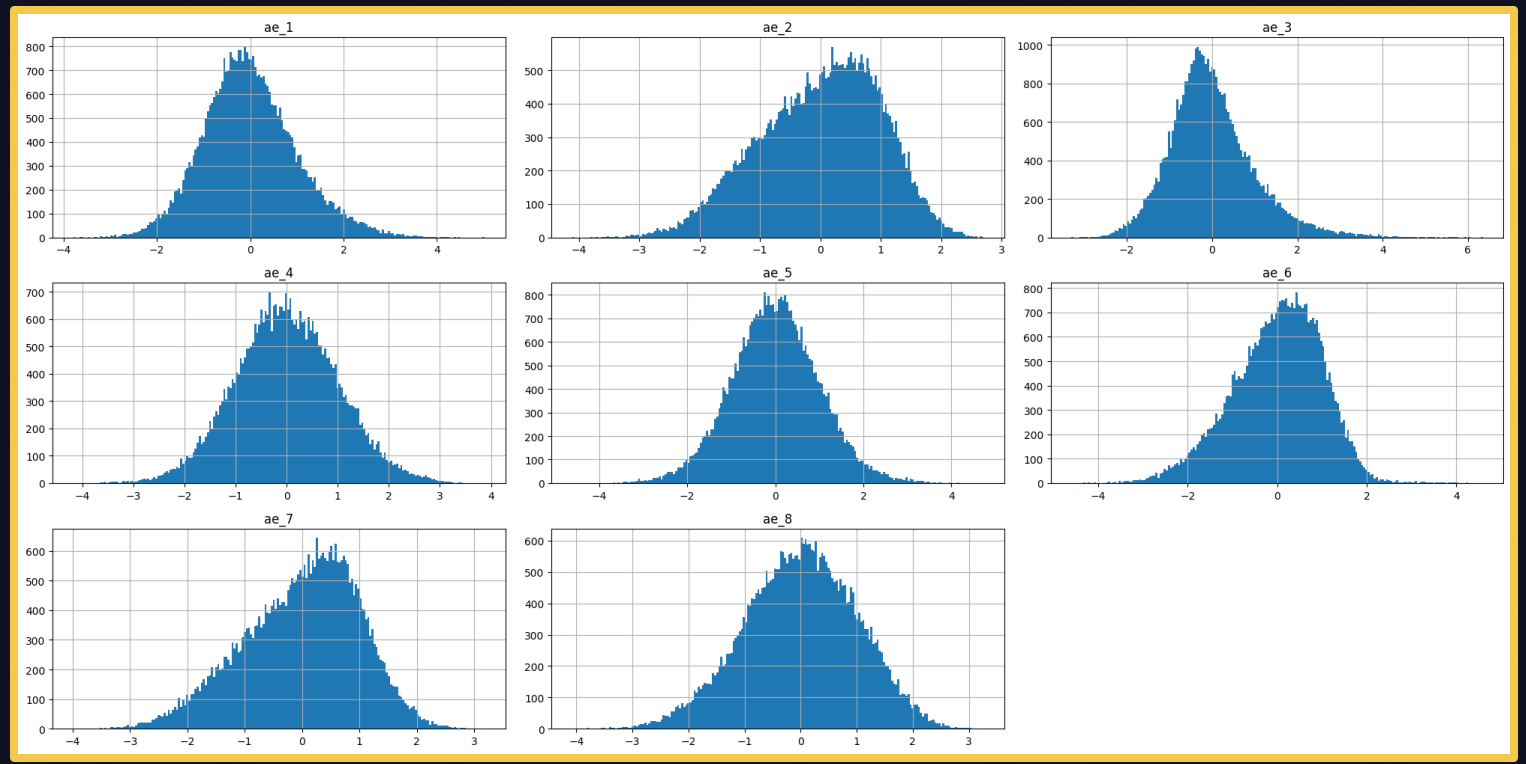
What if we tried to predict if a player's immediate next move will be a blunder? (see Appendix)

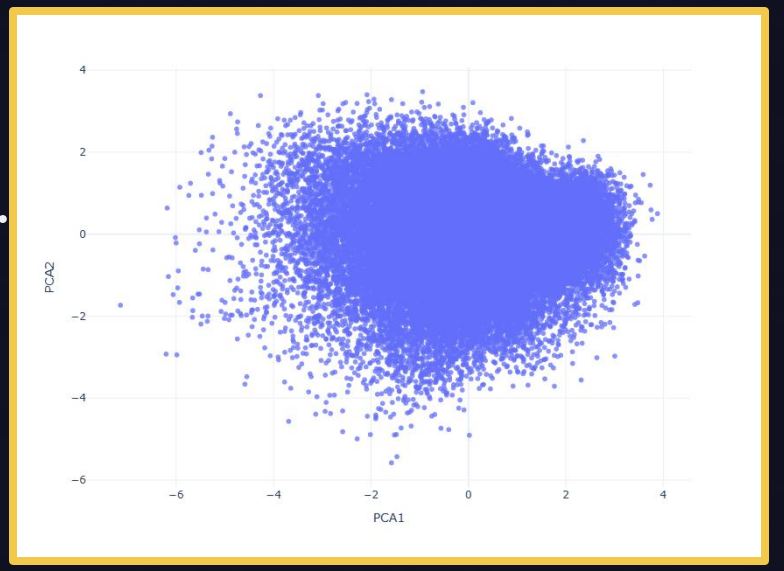
Train on BCE? (see Appendix)

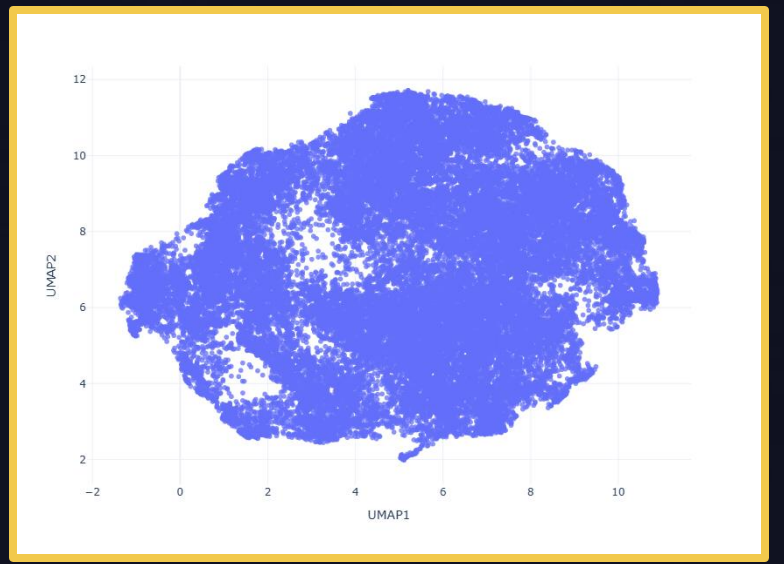
APPENDIX

Attempt: clustering by AutoEncoder

Feed SAN string directly to AE, let *it* derive a 8D latent space.
Result: virtually same structure as the summary features.
Clustering attempts equally as unsuccessful.

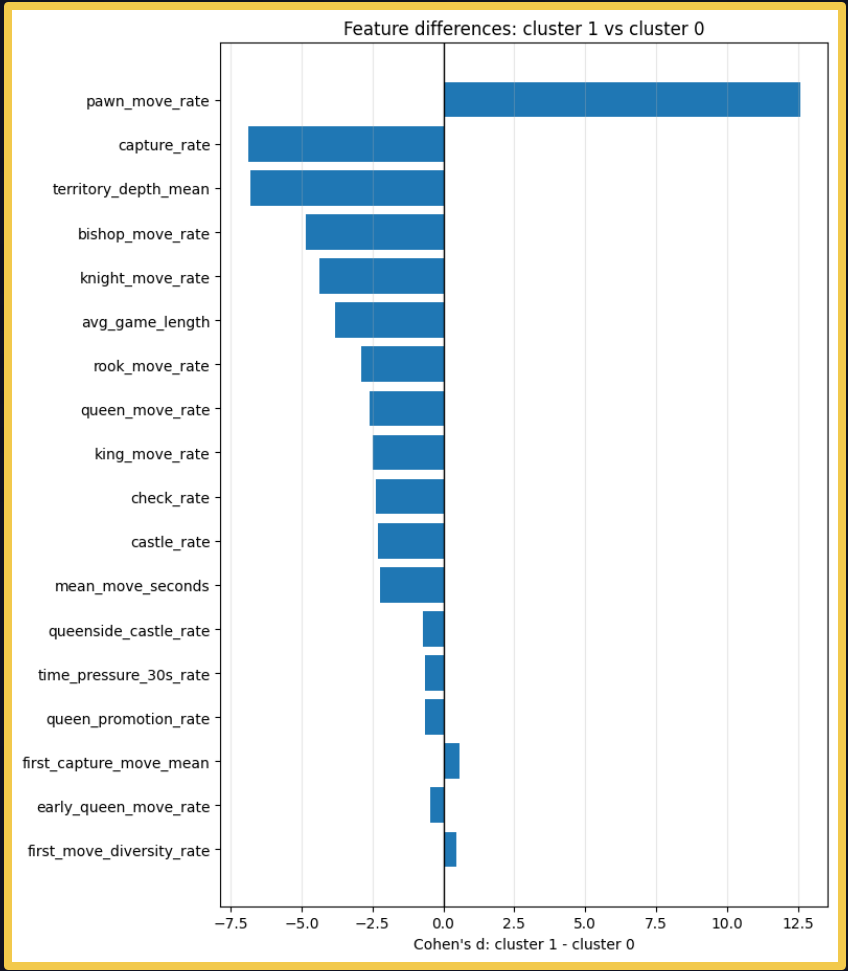


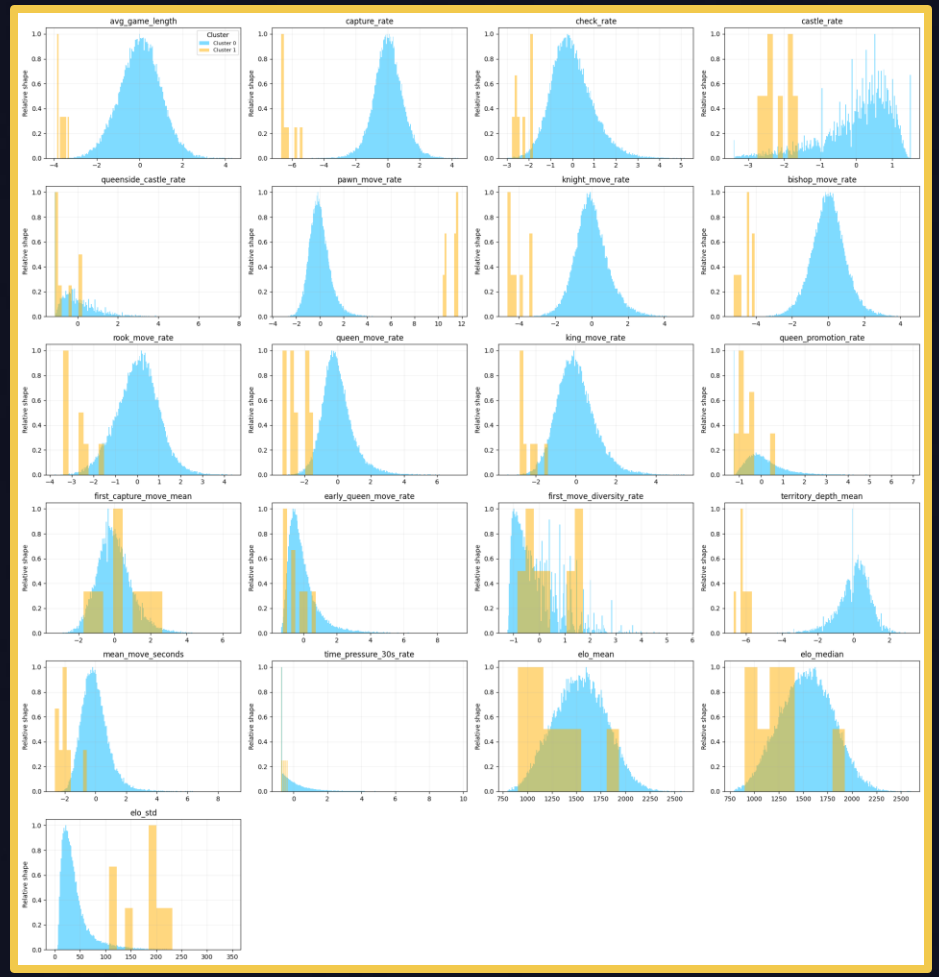




Playstyle clustering – Detected cheats?

DBSCAN: Feature distributions (yellow = “cheaters”, blue =everyone else).





Blunder classification full features list

"regular" (board-derived) features

```
37 candidate features
mover_is_white
fullmove_number
fullmove_number_log
eval_for_mover
eval_before_for_mover
abs_eval_pawns
current_pawn_loss
current_is_mistake
current_is_blunder
current_eval_swing_for_mover
abs_current_eval_swing
phase_progress_current
opening_weight_current
middlegame_weight_current
endgame_weight_current
material_imbalance_for_mover
non_pawn_imbalance_for_mover
total_material_current
total_non_pawn_material_current
both_queens_present_current
no_queens_present_current
clock_seconds_current
log_clock_seconds_current
previous_own_clock_seconds
own_clock_change
previous_own_pawn_loss
previous_own_mistakes
previous_own_blunders
previous_loss_mean_3
previous_loss_max_3
previous_loss_std_3
previous_loss_mean_5
previous_loss_max_5
previous_loss_std_5
previous_loss_mean_10
previous_loss_max_10
previous_loss_std_10
```

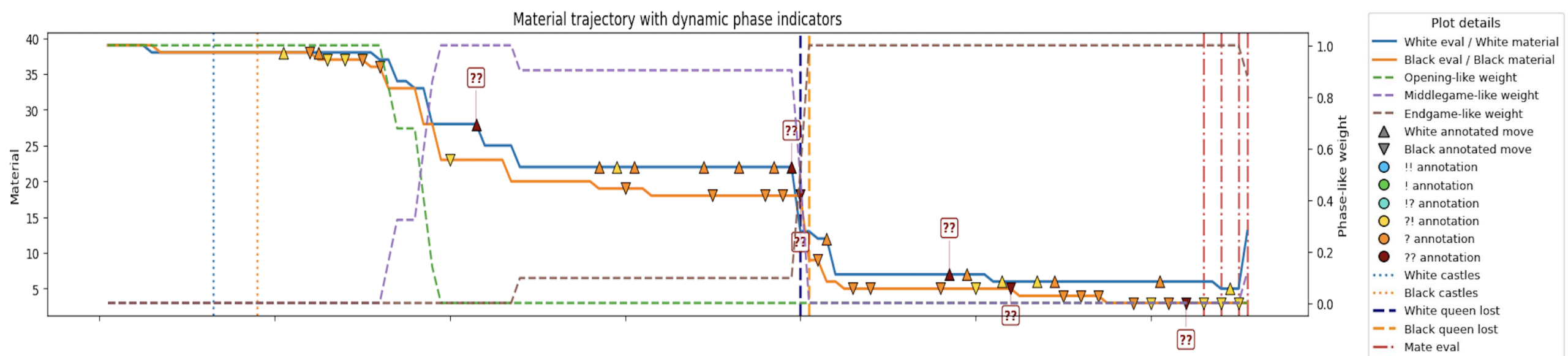
"meta" (from Lichess itself) features

```
No-meta features: 37
With-meta features: 50
Added meta features:
mover_elo
opponent_elo
avg_elo
elo_diff_for_mover
abs_elo_diff
time_base_seconds
time_increment_seconds
log_time_base_seconds
log_time_increment_seconds
has_increment
time_control_category
event_is_rated
event_is_tournament
```

Blunder classification engineered features explanation (non-exhaustive)

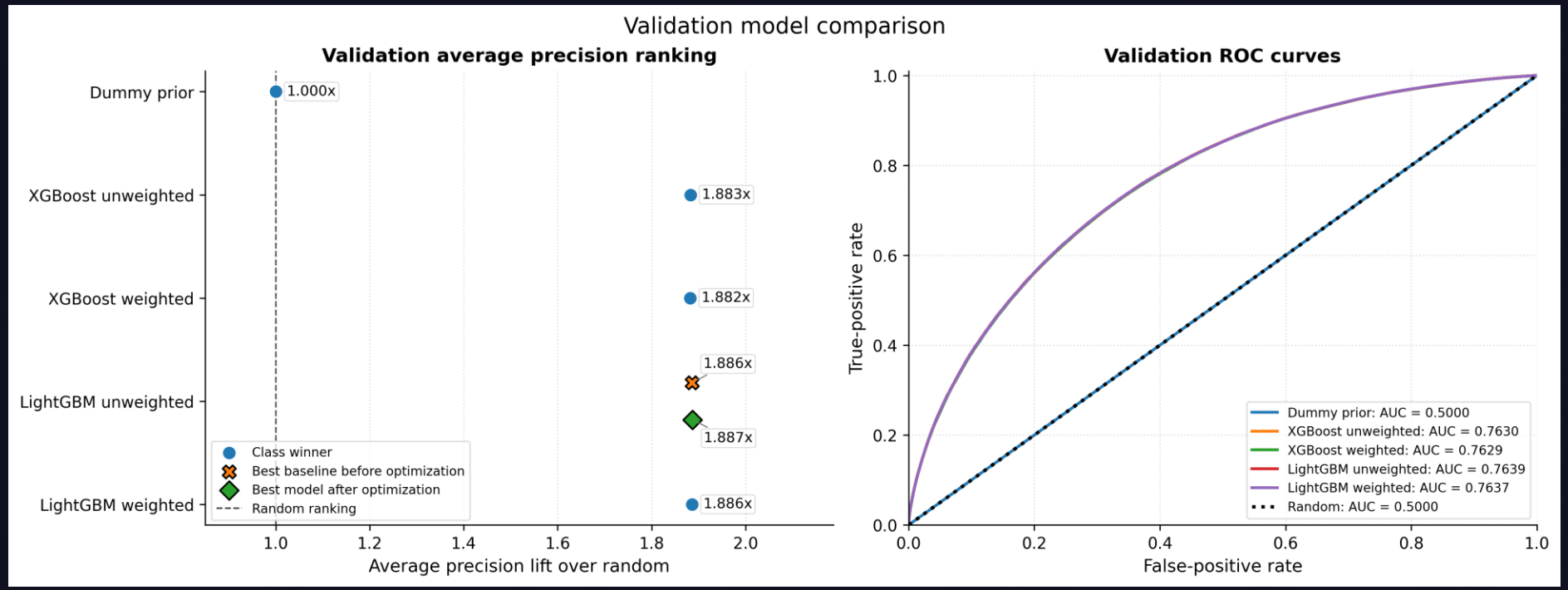
Material explanation: pawn = 1, knight = 3, bishop = 3, rook = 5, queen = 9, king = undefined

- `phase_progress_current` : 1 - (current non pawn material)/(starting non pawn material = 62)
- `material_imbalance_for_mover` : material advantage from the side-to-move's perspective (negative if disadvantage)
- `total_material_current` : total material on the board
- `total_non_pawn_material_current` : total non-pawn material on board
- `(opening/middle/end)game_weight_current` : see example game below
  - `opening_weight_current`=((0.30-`phase_progress_current`)/0.30).clip(0.0,1.0)*`both_queens`, where `both_queens` = 1 if both queens are present
  - `endgame_weight_current`=max{ (`phase_progress_current`-0.45/0.35).clip(0.0,1.0), ((`phase_progress_current`-0.25)/0.35).clip(0.0,1.0)*`no_queens` }, where `no_queens` is 1 if no queens are present
  - `middlegame_weight_current`=1.0-max{`opening_weight_current`,`endgame_weight_current`}

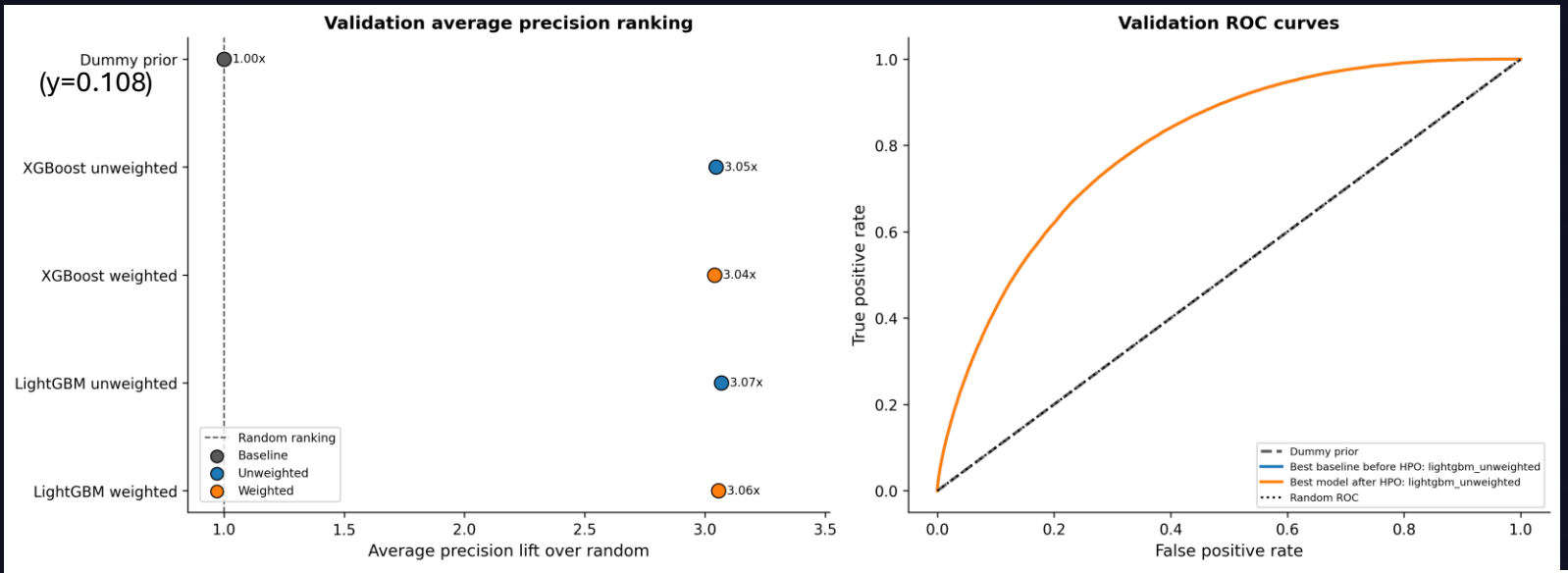


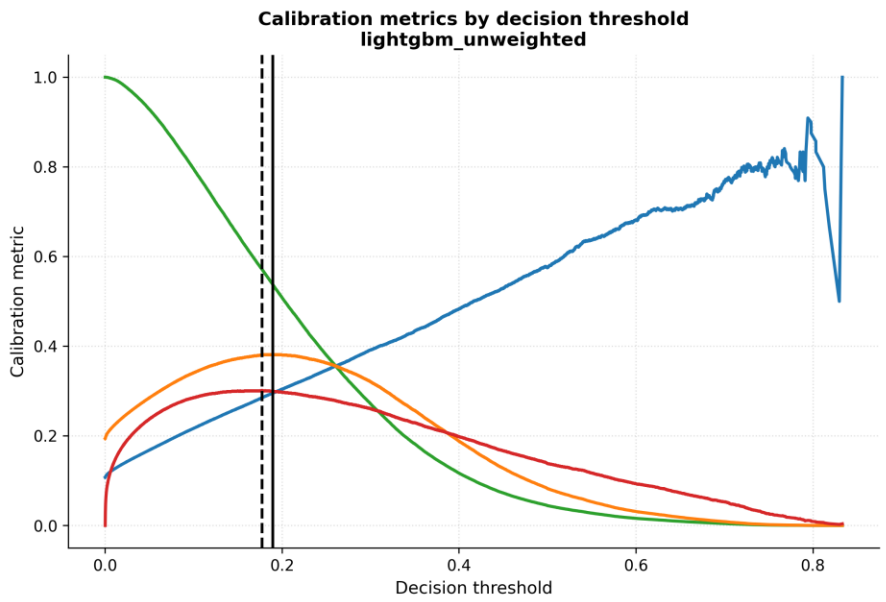
Blunder problem hyperparameterization

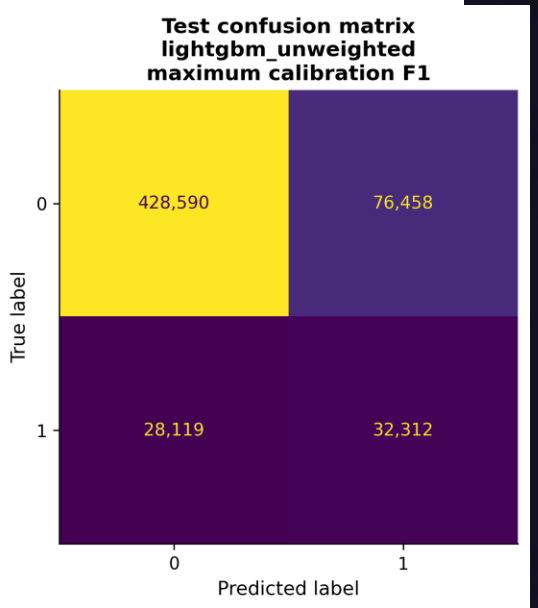
Used Optuna for Bayesian HPO -> little impact on results. (note: only 1 model is optimized in the example fig. below, but this was observed in all blunder classification optimization attempts)



1-move window blunder prediction



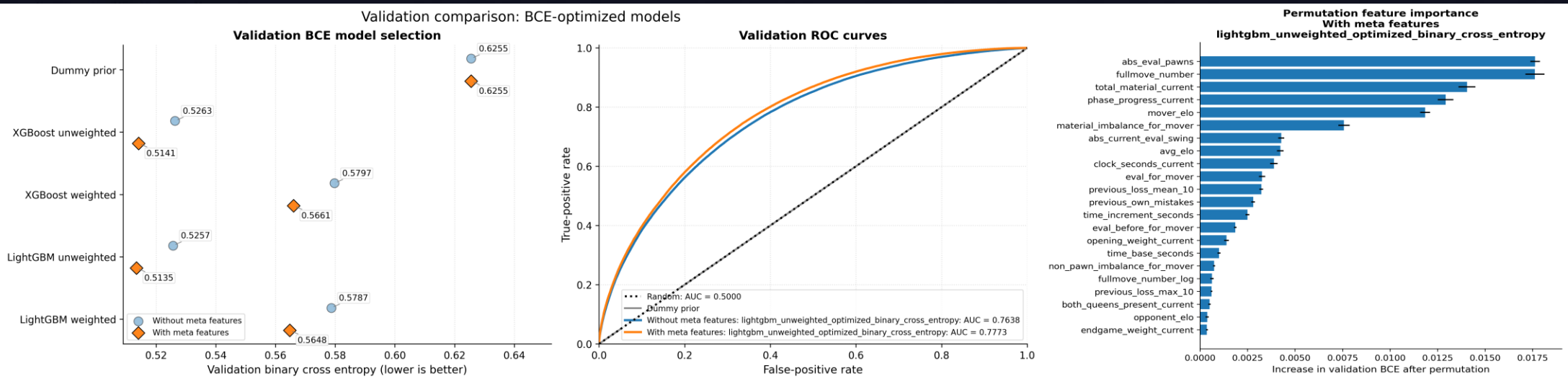


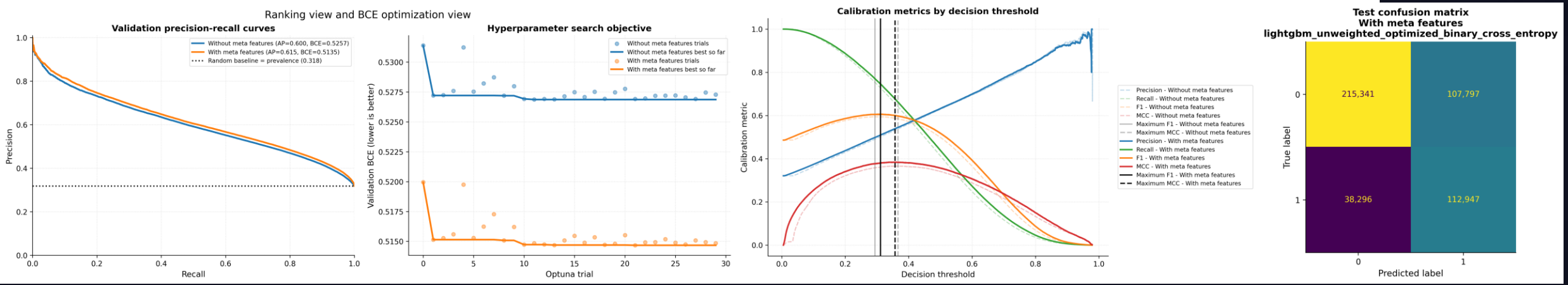


Blunders occur in ca. 10% of moves of the dataset. Calibrated models (trained on regular + meta features, average precision target) can precit blunders with ca. 3x the average precision of a dummy prior. Somewhat surprisingly, weighing the inputs does not give a significant AP lift.

Like the 5-move window case, we choose a decision threshold based on the F1 score. At this threshold, recall > precision, so we tend to predict more false positives than false negatives.

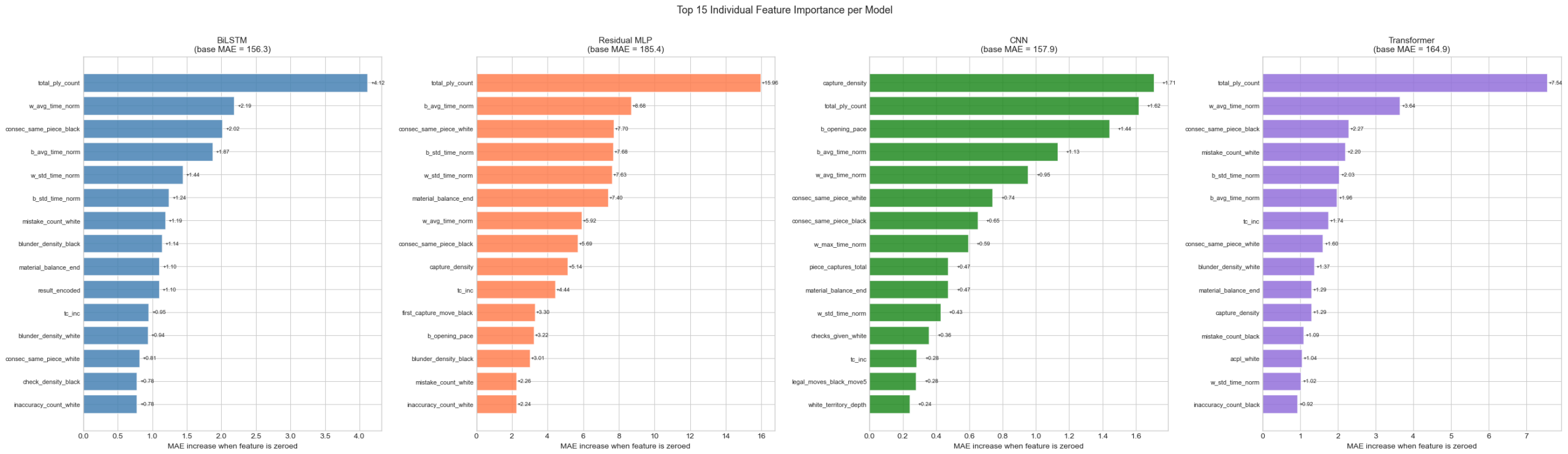
Training blunder classification BDTs on BCE



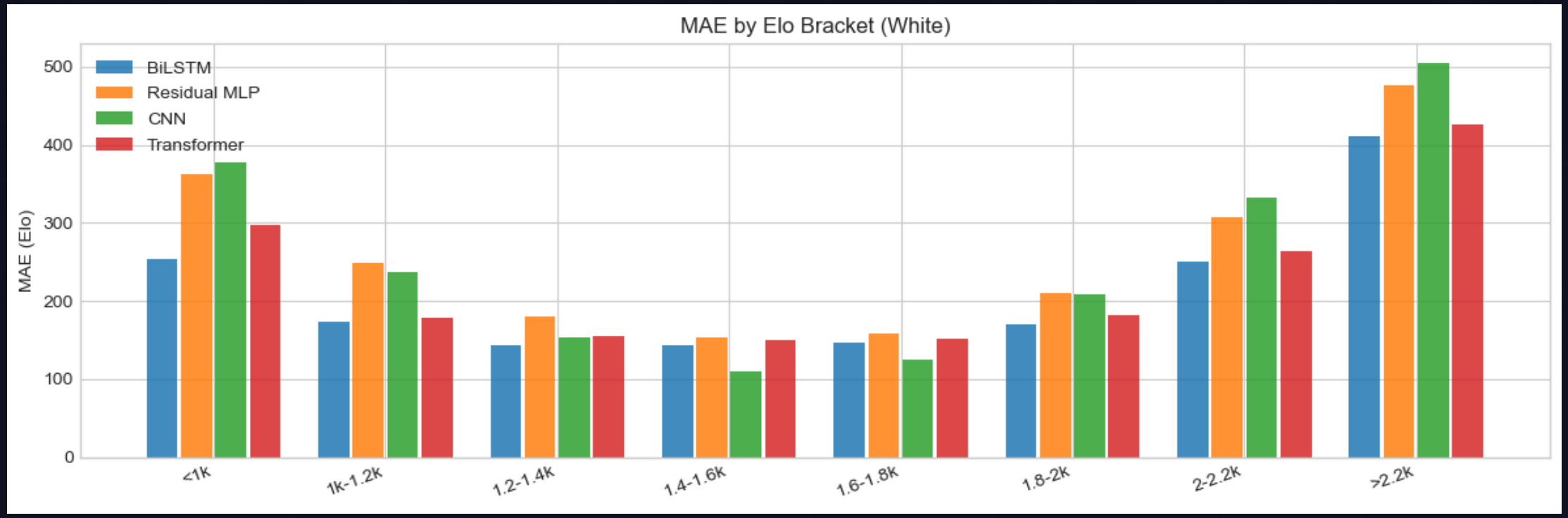


These results are for the 5-move window case trained on minimizing BCE. The validation-precision curves, HPO plot, and calibration metrics plot are for the model with the lowest BCE score: LightGBM unweighted. The confusion matrix threshold was set using the best F1 score, like in the AP case. Here, the unweighted models yield noticably better BCE scores than the weighted cases, and because the imbalance is not so severe here (32%-68%), we proceed with the best (unweighted) model. The shape of the calibration metric curves and the final confusion matrix are not so different from the AP case.

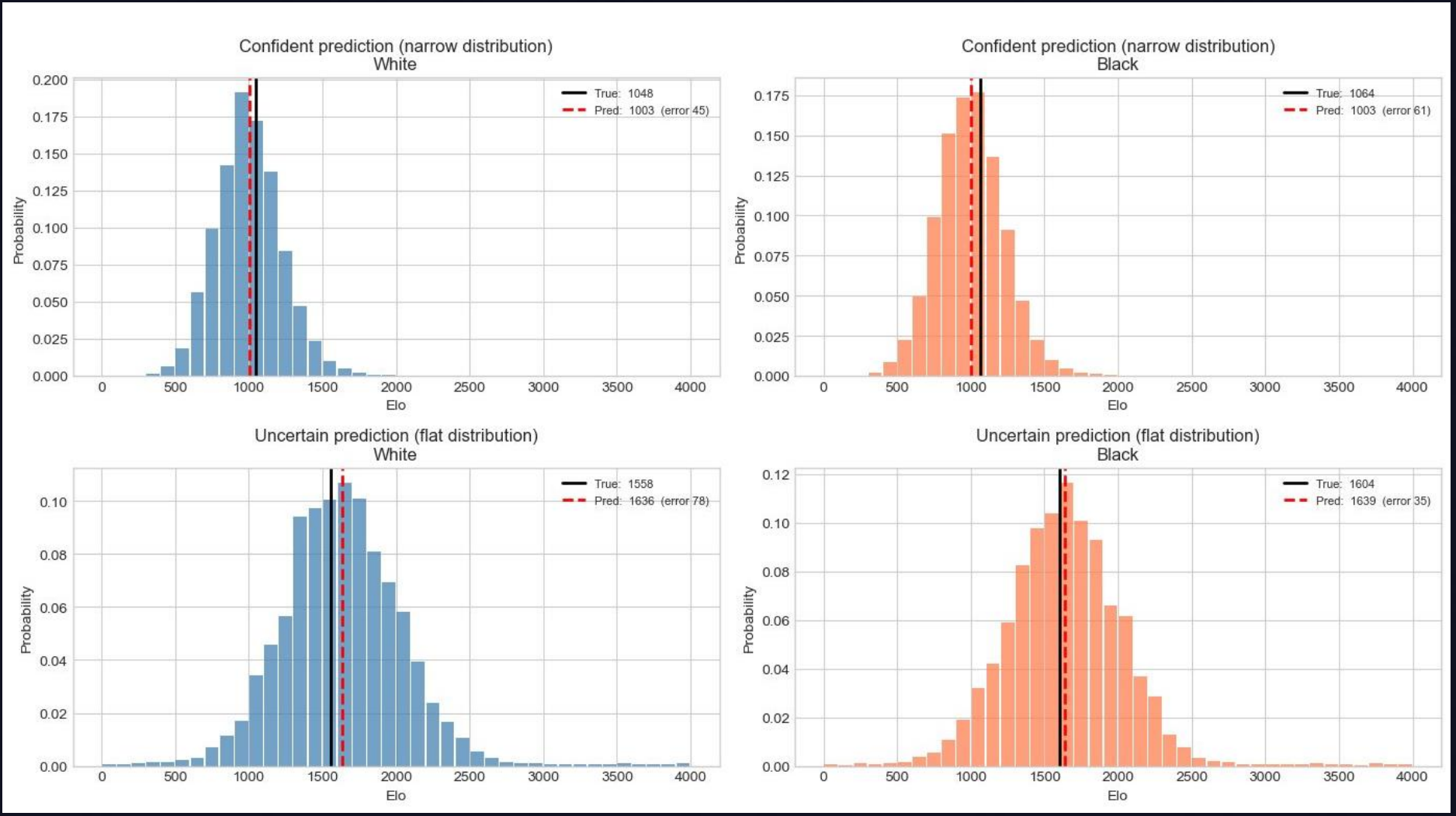
Individual feature importance of Neural network



Neural network: MAE by Elo Bracket



Neural network output



Neural network

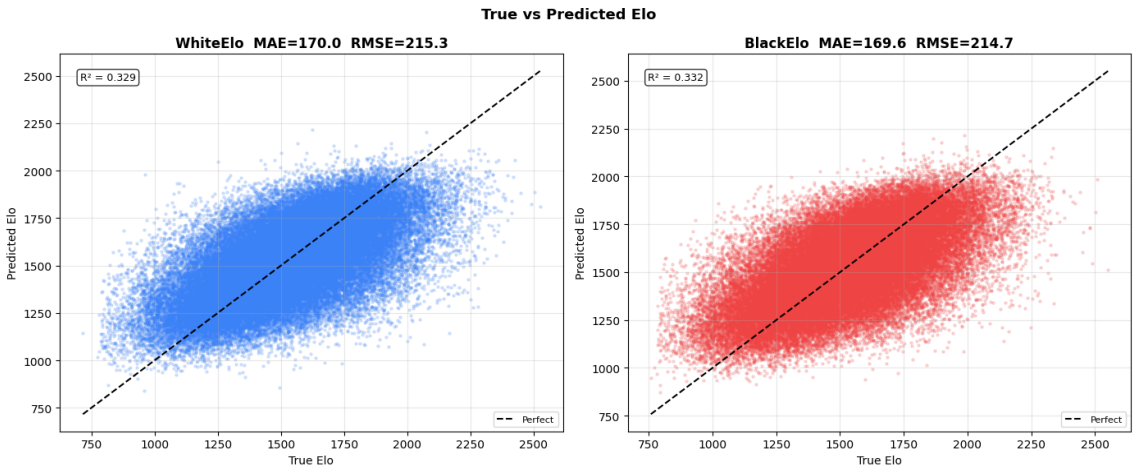
Problem: Seems like The models take an "Easy" route with relying only on move sequence

Our solution: Increase dropout, to reduce very weak regularisation, so the model can does not memories training examples.

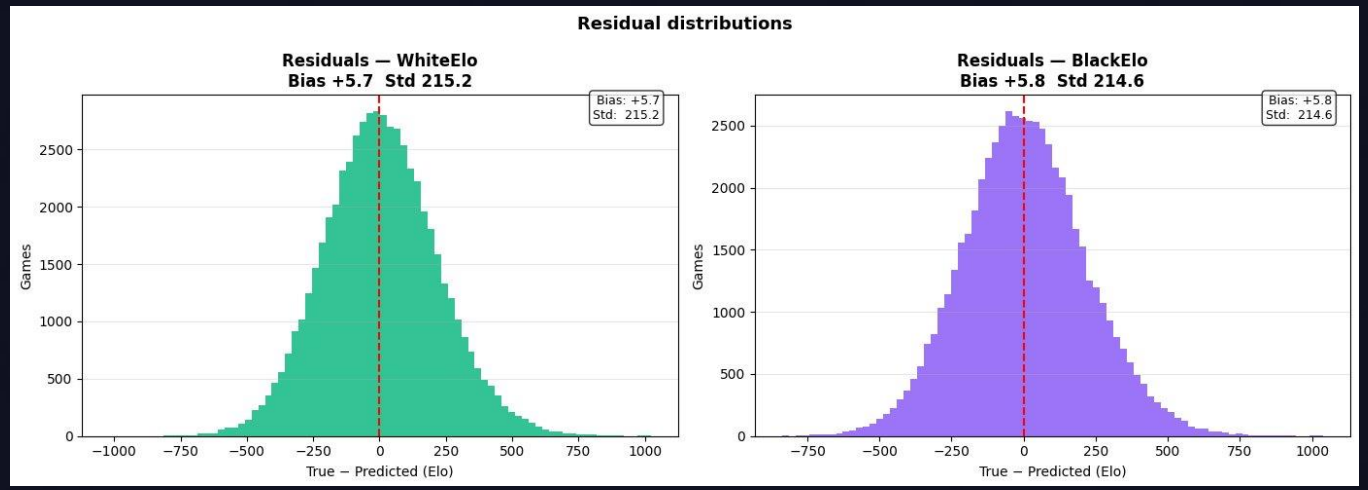
Neural network

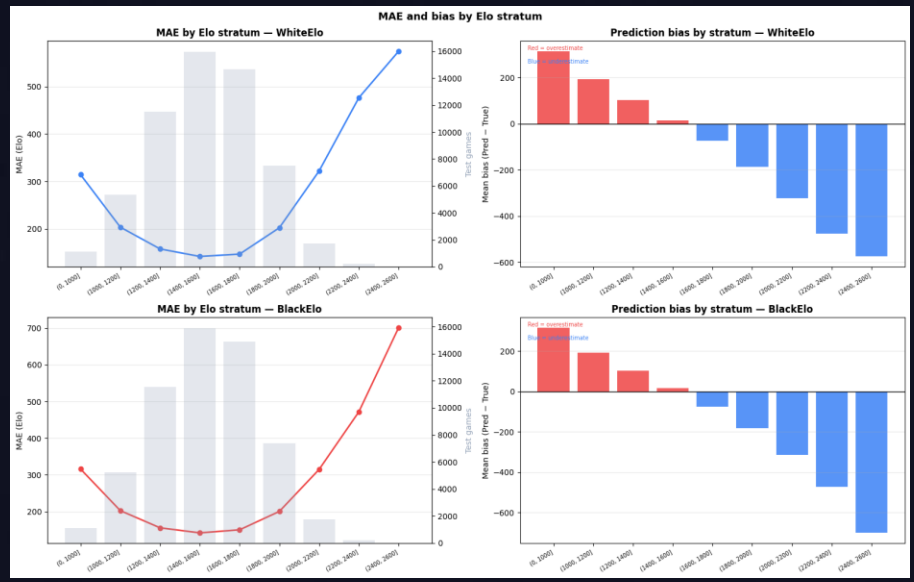
- Low dropout let models take the easy route — rely on sequence and ignore statistics
- Higher dropout forces the model to extract signal from both branches
- Balanced contribution = more robust model that uses all available information
- Transformer exception: attention mechanism so powerful it crowds out static features regardless of dropout => a known phenomenon called feature collapse
- Style group jumped from +25 to +46 MAE for Residual MLP — with proper regularisation, piece activity and opening decisions are the most reliable static signal

Regression BDT – Appendix

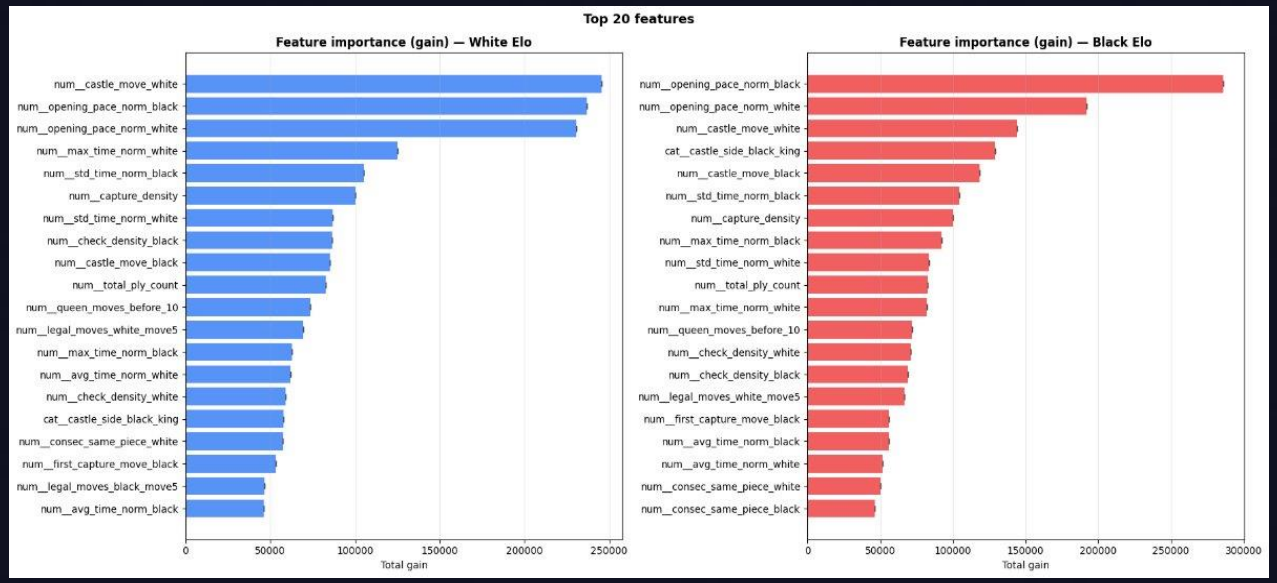


Best model without using player aggregation





Regression BDT – Features and ranking



Category / Sub-Group	Feature Variable Name(s)
1. Board Features	
Structure	total_ply_count material_balance material_balance_end
Checks	checks_given_white, checks_given_black check_density_white, check_density_black
Captures	first_capture_move_white, first_capture_move_black pawn_captures_total, piece_captures_total capture_density
Castling	castle_move_white, castle_move_black
Style	consec_same_piece_white, consec_same_piece_black queen_moves_before_10 white_territory_depth, black_territory_depth promotions, en_passant_captures legal_moves_white_move5, legal_moves_black_move5
2. Clock Features	
White	avg_time_norm_white, std_time_norm_white max_time_norm_white, time_pressure_white opening_pace_norm_white, clock_remaining_norm_white opening_time_norm_white, endgame_time_norm_white time_spent_trend_norm_white
Black	avg_time_norm_black, std_time_norm_black max_time_norm_black, time_pressure_black opening_pace_norm_black, clock_remaining_norm_black opening_time_norm_black, endgame_time_norm_black time_spent_trend_norm_black

Single Game Demo Neural network – actual ELO: 1516

Model	White Elo	Black Elo
BiLSTM	1818	1804
Residual MLP	1833	1736
CNN	1734	1739
Transformer	1784	1788

Challenges for the Neural Networks

Loss, dropout & HPO

- Val loss consistently below train loss => Overfitting, The weighted sampler showing harder examples during training
- Dropout seems very important to stop models from taking an "easy route"
- HPO seemed to get dropout wrong

No Engine Evaluations and Data processing

•Most Lichess games have no [%eval] tags - We would assume that these would be the strongest features by far if available

•Weighing the distribution, amount of data vs. Performance, Tokenizing moves

Future prospect for NN's

- Add Stockfish evaluations
- Try different HPO methods
- Board position features - represent each position as an 8×8 array through a 2D CNN instead of move tokens
- Separate models per time control - blitz and rapid reveal different aspects of player strength
- More recent Lichess data - 2017 database is small by modern standards, rating pool much larger today
- Longer Transformer training on GPU (And generally all models on GPU)

The Models

- **Residual MLP**
  - Only sees the 45 static features : never touches the moves
  - Four residual blocks where each block does output = x + f(x) — skip connections prevent unstable training
  - The baseline: MAE 185 tells us exactly what statistics alone are worth
- **BiLSTM**
  - Embeds each move as a 64-number vector, appends the clock value, feeds into the LSTM step by step
  - Reads forward and backward (bidirectional=True) so each position has context from both directions
  - Attention scores each position, padding gets -1e9 so it gets zero weight, result is a weighted average across all moves
  - Sequence summary (256-d) concatenated with static summary (32-d) before the prediction heads
- **CNN**
  - Three parallel sliding windows of size 3, 5, 7 scan the sequence simultaneously
  - Each window keeps only its strongest activation across the whole game (max pooling): finds the best pattern regardless of where it occurred
  - Fastest to converge, best MAE at 158 — max pooling is particularly robust for this task
- **Transformer**
  - Sinusoidal positional encodings added first — without them the model treats the game as an unordered bag of moves
  - Self-attention lets every move attend directly to every other move — no sequential dependency like the LSTM
  - Mean pooling over all real positions produces the sequence summary
  - Most powerful in theory but needs more data and training time to show its advantage over shorter sequences